

Land Information System

LIS Reference Manual

April 17, 2006



National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, MD 20771
<http://lis.gsfc.nasa.gov>

Contents

I LIS Overview	18
1 Release Notes	19
2 What is the Land Information System?	19
3 The LIS Reference Manual	19
II LIS offline program	20
4 LIS offline program	21
4.0.1 lisdrv (Source File: lisdrv.F90)	21
III LIS core structures and methods	22
5 LIS core structures and methods	23
5.1 Fortran: Module Interface lisdrv_module (Source File: lisdrv_module.F90)	23
5.1.1 Overview	23
5.1.2 lisconfig (Source File: lisdrv_module.F90)	24
5.1.3 lisconfig_offline (Source File: lisdrv_module.F90)	24
5.1.4 lisconfig_coupled (Source File: lisdrv_module.F90)	25
5.1.5 LIS_ticktime (Source File: lisdrv_module.F90)	25
5.1.6 LIS_endofrun (Source File: lisdrv_module.F90)	26
5.1.7 LIS_timeToRunNest (Source File: lisdrv_module.F90)	26
5.1.8 LIS_finalize (Source File: lisdrv_module.F90)	26
5.2 Fortran: Module Interface lis_module (Source File: lis_module.F90)	27
5.3 Fortran: Module Interface grid_module (Source File: grid_module.F90)	31
5.3.1 Overview	31
5.4 Fortran: Module Interface listime_mgr (Source File: listime_mgr.F90)	31
5.4.1 isMonthlyAlarmRinging (Source File: listime_mgr.F90)	32
5.4.2 timemgr_init (Source File: listime_mgr.F90)	32
5.4.3 timemgr_set (Source File: listime_mgr.F90)	32
5.4.4 timemgr_print (Source File: listime_mgr.F90)	33
5.4.5 advance_timestep (Source File: listime_mgr.F90)	33
5.4.6 get_step_size (Source File: listime_mgr.F90)	34
5.4.7 get_nstep (Source File: listime_mgr.F90)	34
5.4.8 get_curr_day (Source File: listime_mgr.F90)	34
5.4.9 get_julhr (Source File: listime_mgr.F90)	35
5.4.10 get_curr_calday (Source File: listime_mgr.F90)	36
5.4.11 is_last_step (Source File: listime_mgr.F90)	36
5.4.12 setMonthlyAlarm (Source File: listime_mgr.F90)	37
5.4.13 isMonthlyAlarmRinging1 (Source File: listime_mgr.F90)	37
5.4.14 isMonthlyAlarmRinging2 (Source File: listime_mgr.F90)	38
5.4.15 setHourlyAlarm (Source File: listime_mgr.F90)	38
5.4.16 isHourlyAlarmRinging (Source File: listime_mgr.F90)	38
5.4.17 computeTimeBookEnds (Source File: listime_mgr.F90)	39
5.4.18 computeMonthlyWeights (Source File: listime_mgr.F90)	39

5.4.19	computeMonthlyWeights (Source File: listime_mgr.F90)	40
5.4.20	date2time (Source File: listime_mgr.F90)	41
5.4.21	time2date (Source File: listime_mgr.F90)	41
5.4.22	tick (Source File: listime_mgr.F90)	42
5.4.23	julhr_date (Source File: listime_mgr.F90)	43
5.4.24	Method	43
5.4.25	tmjul4 (Source File: listime_mgr.F90)	44
5.4.26	Method	44
5.5	Fortran: Module Interface spmdMod (Source File: spmdMod.F90)	44
5.5.1	spmd_init (Source File: spmdMod.F90)	45
5.5.2	spmd_init_offline (Source File: spmdMod.F90)	45
5.5.3	spmd_init_coupled (Source File: spmdMod.F90)	46
5.5.4	spmd_setup (Source File: spmdMod.F90)	46
5.5.5	spmd_finalize (Source File: spmdMod.F90)	46
5.6	Fortran: Module Interface output_metaMod (Source File: output_metaMod.F90)	47
5.7	Fortran: Module Interface LIS_ConfigMod - Implements LIS configuration management (Source File: LIS_ConfigMod.F90)	48
5.7.1	Package Overview	48
5.7.2	Resource Files	48
5.7.3	A Quick Stroll	49
5.7.4	Package History	50
5.7.5	LIS_ConfigGetAttribute	51
5.7.6	LIS_ConfigCreate	51
5.7.7	LIS_ConfigDestroy	51
5.7.8	LIS_ConfigFindLabel	52
5.7.9	LIS_ConfigGetAttribute	52
5.7.10	LIS_ConfigGetAttribute	53
5.7.11	LIS_ConfigGetAttribute	54
5.7.12	LIS_ConfigGetAttribute	54
5.7.13	LIS_ConfigGetAttribute	55
5.7.14	LIS_ConfigGetAttribute	55
5.7.15	LIS_ConfigGetAttribute	56
5.7.16	LIS_ConfigGetAttribute	57
5.7.17	LIS_ConfigGetAttribute	57
5.7.18	LIS_ConfigGetChar	58
5.7.19	LIS_ConfigGetDim	58
5.7.20	LIS_ConfigGetLen	59
5.7.21	LIS_ConfigNextLine	59
5.7.22	LIS_ConfigLoadFile	60
5.7.23	LIS_ConfigLoadFile_1proc	60
5.8	Fortran: Module Interface domain_module.F90 (Source File: domain_module.F90)	61
5.8.1	Overview	61
5.8.2	LIS_domain_init (Source File: domain_module.F90)	61
5.8.3	LIS_domain_finalize (Source File: domain_module.F90)	62
5.9	Fortran: Module Interface lsm_module (Source File: lsm_module.F90)	62
5.9.1	Overview	62
5.9.2	LIS_force2tile (Source File: lsm_module.F90)	63
5.9.3	LIS_lsm_init (Source File: lsm_module.F90)	63
5.9.4	LIS_setuplsm (Source File: lsm_module.F90)	64
5.9.5	LIS_lsm_main (Source File: lsm_module.F90)	64
5.9.6	LIS_readrestart (Source File: lsm_module.F90)	65

5.9.7	LIS_lsm_output (Source File: lsm_module.F90)	65
5.9.8	setLSMDynparams (Source File: lsm_module.F90)	65
5.9.9	LIS_force2tile_offline (Source File: lsm_module.F90)	66
5.9.10	LIS_force2tile_coupled (Source File: lsm_module.F90)	67
5.9.11	LIS_writerestart (Source File: lsm_module.F90)	67
5.9.12	LIS_lsm_setexport (Source File: lsm_module.F90)	68
5.9.13	LIS_lsm_finalize (Source File: lsm_module.F90)	68
5.10	Fortran: Module Interface baseforcing_module (Source File: baseforcing_module.F90)	68
5.10.1	Overview	69
5.10.2	LIS_baseforcing_init (Source File: baseforcing_module.F90)	69
5.10.3	LIS_get_base_forcing (Source File: baseforcing_module.F90)	70
5.10.4	LIS_baseforcing_finalize (Source File: baseforcing_module.F90)	71
5.10.5	allocate_forcing_mem (Source File: baseforcing_module.F90)	71
5.11	Fortran: Module Interface suppforcing_module (Source File: suppforcing_module.F90)	71
5.11.1	Overview	71
5.11.2	LIS_suppforcing_init (Source File: suppforcing_module.F90)	72
5.11.3	LIS_get_supp_forcing (Source File: suppforcing_module.F90)	72
5.11.4	LIS_suppforcing_finalize (Source File: suppforcing_module.F90)	73
5.12	Fortran: Module Interface dataassim_module (Source File: dataassim_module.F90)	73
5.12.1	Overview	73
5.12.2	LIS_dataassim_init (Source File: dataassim_module.F90)	74
5.12.3	LIS_dataassim_finalize (Source File: dataassim_module.F90)	75
5.13	Fortran: Module Interface param_module (Source File: param_module.F90)	75
5.13.1	Overview	75
5.13.2	LIS_param_init (Source File: param_module.F90)	76
5.13.3	LIS_setDynparams (Source File: param_module.F90)	77
5.13.4	LIS_param_finalize (Source File: param_module.F90)	77
5.14	Fortran: Module Interface albedo_module (Source File: albedo_module.F90)	78
5.14.1	Overview	78
5.14.2	albedo_setup (Source File: albedo_module.F90)	79
5.14.3	read_mxsnalb (Source File: albedo_module.F90)	79
5.14.4	read_albedo (Source File: albedo_module.F90)	80
5.14.5	albedo_finalize (Source File: albedo_module.F90)	80
5.15	Fortran: Module Interface gfrac_module (Source File: gfrac_module.F90)	80
5.15.1	Overview	81
5.15.2	greenness_setup (Source File: gfrac_module.F90)	81
5.15.3	read_greenness (Source File: gfrac_module.F90)	81
5.15.4	greenness_finalize (Source File: gfrac_module.F90)	82
5.16	Fortran: Module Interface lai_module (Source File: lai_module.F90)	82
5.16.1	Overview	82
5.16.2	lai_setup (Source File: lai_module.F90)	83
5.16.3	read_lai (Source File: lai_module.F90)	83
5.16.4	lai_finalize (Source File: lai_module.F90)	84
5.17	Fortran: Module Interface sai_module (Source File: sai_module.F90)	84
5.17.1	Overview	84
5.17.2	sai_setup (Source File: sai_module.F90)	84
5.17.3	read_sai (Source File: sai_module.F90)	85
5.17.4	sai_finalize (Source File: sai_module.F90)	85
5.18	Fortran: Module Interface snow_module (Source File: snow_module.F90)	86
5.18.1	snow_setup (Source File: snow_module.F90)	86
5.18.2	read_snow (Source File: snow_module.F90)	86

5.18.3	snow_finalize (Source File: snow_module.F90)	87
5.19	Fortran: Module Interface soils_module (Source File: soils_module.F90)	87
5.19.1	Overview	87
5.19.2	soils_setup (Source File: soils_module.F90)	88
5.19.3	read_soils (Source File: soils_module.F90)	88
5.19.4	soils_finalize (Source File: soils_module.F90)	89
5.19.5	check_error (Source File: check_error.F90)	89
5.20	Fortran: Module Interface constantsmod.F90 (Source File: constantsMod.F90)	90
5.21	Fortran: Module Interface drv_output_mod (Source File: drv_output_mod.F90)	91
5.21.1	Overview	91
5.21.2	drv_writevar_bin (Source File: drv_output_mod.F90)	91
5.21.3	drv_writevar_restart (Source File: drv_output_mod.F90)	92
5.21.4	drv_readvar_restart (Source File: drv_output_mod.F90)	92
5.21.5	drv_writevar_grib (Source File: drv_output_mod.F90)	93
5.21.6	drv_writevar_netcdf (Source File: drv_output_mod.F90)	93
5.21.7	writevar_bin_withstats_real (Source File: drv_output_mod.F90)	93
5.21.8	writevar_bin_withstats_int (Source File: drv_output_mod.F90)	94
5.21.9	writevar_bin_real (Source File: drv_output_mod.F90)	95
5.21.10	writevar_bin_int (Source File: drv_output_mod.F90)	95
5.21.11	writevar_restart_real (Source File: drv_output_mod.F90)	96
5.21.12	writevar_restart_int (Source File: drv_output_mod.F90)	97
5.21.13	readvar_restart_real (Source File: drv_output_mod.F90)	97
5.21.14	readvar_restart_int (Source File: drv_output_mod.F90)	98
5.21.15	writevar_netcdf_2d (Source File: drv_output_mod.F90)	98
5.21.16	drv_writevar_netcdf_3d (Source File: drv_output_mod.F90)	99
5.21.17	writevar_grib_real (Source File: drv_output_mod.F90)	100
5.21.18	writevar_grib_withstats_real (Source File: drv_output_mod.F90)	100
5.21.19	drv_tile2grid (Source File: drv_output_mod.F90)	101
5.21.20	endrun (Source File: endrun.F90)	102
5.22	Fortran: Module Interface gswp_module (Source File: gswp_module.F90)	102
5.22.1	getgswp_monindex (Source File: gswp_module.F90)	103
5.22.2	getgswp_timeindex (Source File: gswp_module.F90)	103
5.22.3	lapseRateCorrection (Source File: lapseRateCorrection.F90)	103
5.23	Fortran: Module Interface lis_fileIOMod (Source File: lis_fileIOMod.F90)	104
5.23.1	putget (Source File: lis_fileIOMod.F90)	104
5.23.2	lis_set_filename (Source File: lis_fileIOMod.F90)	105
5.23.3	lis_open_file (Source File: lis_fileIOMod.F90)	105
5.23.4	lis_read_file (Source File: lis_fileIOMod.F90)	106
5.23.5	retrieve_data (Source File: lis_fileIOMod.F90)	107
5.23.6	retrieve_script (Source File: lis_fileIOMod.F90)	108
5.23.7	create_output_directory (Source File: lis_fileIOMod.F90)	108
5.23.8	create_output_filename (Source File: lis_fileIOMod.F90)	109
5.23.9	create_restart_filename (Source File: lis_fileIOMod.F90)	109
5.23.10	create_stats_filename (Source File: lis_fileIOMod.F90)	110
5.23.11	putget_int (Source File: lis_fileIOMod.F90)	110
5.23.12	putget_real (Source File: lis_fileIOMod.F90)	111
5.24	Fortran: Module Interface lis_logmod (Source File: lis_logmod.F90)	112
5.24.1	lis_log_msg (Source File: lis_logmod.F90)	113
5.24.2	lis_abort (Source File: lis_logmod.F90)	113
5.24.3	lis_alert (Source File: lis_logmod.F90)	114
5.24.4	lis_flush (Source File: lis_logmod.F90)	114

5.24.5	makepsn (Source File: makepsn.F90)	114
5.24.6	microMetCorrection (Source File: microMetCorrection.F90)	115
5.25	Fortran: Module Interface mpishorthand.F90 (Source File: mpishorthand.F90)	115
5.26	Fortran: Module Interface precision (Source File: precision.F90)	116
5.26.1	readcard (Source File: readcard.F90)	116
5.26.2	readkpds (Source File: readkpds.F90)	117
5.26.3	soiltype (Source File: soiltype.F90)	118
5.26.4	stats (Source File: stats.F90)	119
5.26.5	zterp (Source File: zterp.F90)	119
5.26.6	coszenith (Source File: zterp.F90)	121
5.26.7	localtime (Source File: zterp.F90)	121
5.27	Fortran: Module Interface LIS_alb_FTable (Source File: LIS_alb_FTable.c)	122
5.27.1	registerreadmxsnoalb (Source File: LIS_alb_FTable.c)	122
5.27.2	readmxsnoalb (Source File: LIS_alb_FTable.c)	122
5.27.3	registerreadalbedo (Source File: LIS_alb_FTable.c)	122
5.27.4	readalbedo (Source File: LIS_alb_FTable.c)	123
5.28	Fortran: Module Interface LIS_dataassim_FTable (Source File: LIS_dataassim_FTable.c)	123
5.28.1	registerdaobssetup (Source File: LIS_dataassim_FTable.c)	123
5.28.2	dataobssetup (Source File: LIS_dataassim_FTable.c)	123
5.28.3	registerreadobs (Source File: LIS_dataassim_FTable.c)	124
5.28.4	readobservations (Source File: LIS_dataassim_FTable.c)	124
5.28.5	registerdasetup (Source File: LIS_dataassim_FTable.c)	124
5.28.6	dataassimsetup (Source File: LIS_dataassim_FTable.c)	124
5.28.7	registerfrcst (Source File: LIS_dataassim_FTable.c)	125
5.28.8	forecast (Source File: LIS_dataassim_FTable.c)	125
5.28.9	registerassim (Source File: LIS_dataassim_FTable.c)	125
5.28.10	assimilate (Source File: LIS_dataassim_FTable.c)	126
5.28.11	registerdafinalize (Source File: LIS_dataassim_FTable.c)	126
5.28.12	dafinalize (Source File: LIS_dataassim_FTable.c)	126
5.29	Fortran: Module Interface LIS_domain_FTable (Source File: LIS_domain_FTable.c)	126
5.29.1	registerdomain (Source File: LIS_domain_FTable.c)	127
5.29.2	makedomain (Source File: LIS_domain_FTable.c)	127
5.29.3	registerinput (Source File: LIS_domain_FTable.c)	127
5.29.4	readinput (Source File: LIS_domain_FTable.c)	127
5.30	Fortran: Module Interface LIS_forcing_FTable (Source File: LIS_forcing_FTable.c)	128
5.30.1	registerget (Source File: LIS_forcing_FTable.c)	128
5.30.2	retrieveforcing (Source File: LIS_forcing_FTable.c)	128
5.30.3	registertimeinterp (Source File: LIS_forcing_FTable.c)	128
5.30.4	timeinterp (Source File: LIS_forcing_FTable.c)	129
5.30.5	registerdefinative (Source File: LIS_forcing_FTable.c)	129
5.30.6	definative (Source File: LIS_forcing_FTable.c)	129
5.30.7	registerforcingfinal (Source File: LIS_forcing_FTable.c)	129
5.30.8	forcingfinalize (Source File: LIS_forcing_FTable.c)	130
5.31	Fortran: Module Interface LIS_gfrac_FTable (Source File: LIS_gfrac_FTable.c)	130
5.31.1	registerreadgfrac (Source File: LIS_gfrac_FTable.c)	130
5.31.2	readgfrac (Source File: LIS_gfrac_FTable.c)	130
5.32	Fortran: Module Interface LIS_laisai_FTable (Source File: LIS_laisai_FTable.c)	131
5.32.1	registerreadlai (Source File: LIS_laisai_FTable.c)	131
5.32.2	readlai (Source File: LIS_laisai_FTable.c)	131
5.32.3	registerreadsai (Source File: LIS_laisai_FTable.c)	132
5.32.4	readsai (Source File: LIS_laisai_FTable.c)	132

5.33	Fortran: Module Interface LIS_landcover_FTable (Source File: LIS_landcover_FTable.c)	132
5.33.1	registerreadlc (Source File: LIS_landcover_FTable.c)	132
5.33.2	readlandcover (Source File: LIS_landcover_FTable.c)	133
5.33.3	registerreadmask (Source File: LIS_landcover_FTable.c)	133
5.33.4	readlandmask (Source File: LIS_landcover_FTable.c)	133
5.34	Fortran: Module Interface LIS_lsm_FTable (Source File: LIS_lsm_FTable.c)	134
5.34.1	registerlsmmini (Source File: LIS_lsm_FTable.c)	134
5.34.2	lsmmini (Source File: LIS_lsm_FTable.c)	134
5.34.3	registerlsmrun (Source File: LIS_lsm_FTable.c)	134
5.34.4	lsmrun (Source File: LIS_lsm_FTable.c)	134
5.34.5	registerlsmfinalize (Source File: LIS_lsm_FTable.c)	135
5.34.6	lsmfinalize (Source File: LIS_lsm_FTable.c)	135
5.34.7	registerlsmsetup (Source File: LIS_lsm_FTable.c)	135
5.34.8	lsmsetup (Source File: LIS_lsm_FTable.c)	135
5.34.9	registerlsmrestart (Source File: LIS_lsm_FTable.c)	136
5.34.10	lsmrestart (Source File: LIS_lsm_FTable.c)	136
5.34.11	registerlsmoutput (Source File: LIS_lsm_FTable.c)	136
5.34.12	lsmoutput (Source File: LIS_lsm_FTable.c)	136
5.34.13	registerlsmf2t (Source File: LIS_lsm_FTable.c)	137
5.34.14	lsmf2t (Source File: LIS_lsm_FTable.c)	137
5.34.15	registerlsmwrst (Source File: LIS_lsm_FTable.c)	137
5.34.16	lsmwrst (Source File: LIS_lsm_FTable.c)	138
5.34.17	registergetstatevar (Source File: LIS_lsm_FTable.c)	138
5.34.18	getstatevar (Source File: LIS_lsm_FTable.c)	138
5.34.19	registersetstatevar (Source File: LIS_lsm_FTable.c)	139
5.34.20	setstatevar (Source File: LIS_lsm_FTable.c)	139
5.34.21	registergetnpr (Source File: LIS_lsm_FTable.c)	139
5.34.22	getnpr (Source File: LIS_lsm_FTable.c)	140
5.34.23	registerlsmreset (Source File: LIS_lsm_FTable.c)	140
5.34.24	lsmreset (Source File: LIS_lsm_FTable.c)	140
5.34.25	registerobstransform (Source File: LIS_lsm_FTable.c)	140
5.34.26	obstransform (Source File: LIS_lsm_FTable.c)	141
5.34.27	registerlsmexport (Source File: LIS_lsm_FTable.c)	141
5.34.28	lsmsetexport (Source File: LIS_lsm_FTable.c)	141
5.34.29	registerlsmdynsetup (Source File: LIS_lsm_FTable.c)	142
5.34.30	lsmdynsetup (Source File: LIS_lsm_FTable.c)	142
5.35	Fortran: Module Interface LIS_perturb_FTable (Source File: LIS_perturb_FTable.c)	142
5.35.1	registerperturbsetup (Source File: LIS_perturb_FTable.c)	142
5.35.2	perturbinit (Source File: LIS_perturb_FTable.c)	143
5.35.3	registerperturb (Source File: LIS_perturb_FTable.c)	143
5.35.4	perturb (Source File: LIS_perturb_FTable.c)	143
5.35.5	registergetpertforcing (Source File: LIS_perturb_FTable.c)	143
5.35.6	getpertforcing (Source File: LIS_perturb_FTable.c)	144
5.36	Fortran: Module Interface LIS_runmode_FTable (Source File: LIS_runmode_FTable.c)	144
5.36.1	registerlisinit (Source File: LIS_runmode_FTable.c)	144
5.36.2	lisinit (Source File: LIS_runmode_FTable.c)	144
5.36.3	registerlisrun (Source File: LIS_runmode_FTable.c)	145
5.36.4	lisrun (Source File: LIS_runmode_FTable.c)	145
5.36.5	registerlisfinalize (Source File: LIS_runmode_FTable.c)	145
5.36.6	lisfinalize (Source File: LIS_runmode_FTable.c)	145
5.37	Fortran: Module Interface LIS_soils_FTable (Source File: LIS_soils_FTable.c)	146

5.37.1	registerreadsand (Source File: LIS_soils_FTable.c)	146
5.37.2	readsand (Source File: LIS_soils_FTable.c)	146
5.37.3	registerreadclay (Source File: LIS_soils_FTable.c)	146
5.37.4	readclay (Source File: LIS_soils_FTable.c)	147
5.37.5	registerreadsilt (Source File: LIS_soils_FTable.c)	147
5.37.6	readsilt (Source File: LIS_soils_FTable.c)	147
5.37.7	registerreadtexture (Source File: LIS_soils_FTable.c)	148
5.37.8	readsoiltexture (Source File: LIS_soils_FTable.c)	148
5.37.9	registerreadporosity (Source File: LIS_soils_FTable.c)	148
5.37.10	readporosity (Source File: LIS_soils_FTable.c)	149
5.37.11	registerreadpsisat (Source File: LIS_soils_FTable.c)	149
5.37.12	readpsisat (Source File: LIS_soils_FTable.c)	149
5.37.13	registerreadksat (Source File: LIS_soils_FTable.c)	150
5.37.14	readksat (Source File: LIS_soils_FTable.c)	150
5.37.15	registerreadbexp (Source File: LIS_soils_FTable.c)	150
5.37.16	readbexp (Source File: LIS_soils_FTable.c)	151
5.37.17	registerreadquartz (Source File: LIS_soils_FTable.c)	151
5.37.18	readquartz (Source File: LIS_soils_FTable.c)	151
5.37.19	registerreadcolor (Source File: LIS_soils_FTable.c)	152
5.37.20	readcolor (Source File: LIS_soils_FTable.c)	152
5.38	Fortran: Module Interface LIS_suppforcing_FTable (Source File: LIS_suppforcing_FTable.c)	152
5.38.1	registerdefinenativesupp (Source File: LIS_suppforcing_FTable.c)	152
5.38.2	definenativesupp (Source File: LIS_suppforcing_FTable.c)	153
5.38.3	registerreadsupp (Source File: LIS_suppforcing_FTable.c)	153
5.38.4	getsuppforc (Source File: LIS_suppforcing_FTable.c)	153
5.38.5	registersuppti (Source File: LIS_suppforcing_FTable.c)	153
5.38.6	timeinterpsupp (Source File: LIS_suppforcing_FTable.c)	154
5.38.7	registersuppforcingfinal (Source File: LIS_suppforcing_FTable.c)	154
5.38.8	suppforcingfinalize (Source File: LIS_suppforcing_FTable.c)	154
5.39	Fortran: Module Interface LIS_tbot_FTable (Source File: LIS_tbot_FTable.c)	155
5.39.1	registerreadtbot (Source File: LIS_tbot_FTable.c)	155
5.39.2	readtbot (Source File: LIS_tbot_FTable.c)	155
5.40	Fortran: Module Interface LIS_topo_FTable (Source File: LIS_topo_FTable.c)	155
5.40.1	registerreadelev (Source File: LIS_topo_FTable.c)	155
5.40.2	readelev (Source File: LIS_topo_FTable.c)	156
5.40.3	registerreadslope (Source File: LIS_topo_FTable.c)	156
5.40.4	readslope (Source File: LIS_topo_FTable.c)	156
5.40.5	registerreadaspect (Source File: LIS_topo_FTable.c)	157
5.40.6	readaspect (Source File: LIS_topo_FTable.c)	157
5.40.7	registerreadcurv (Source File: LIS_topo_FTable.c)	157
5.40.8	readcurv (Source File: LIS_topo_FTable.c)	158
5.41	Fortran: Module Interface LIS_snowd_FTable (Source File: LIS_snowd_FTable.c)	158
5.41.1	registerreadsnowdepth (Source File: LIS_snowd_FTable.c)	158
5.41.2	readsnowdepth (Source File: LIS_snowd_FTable.c)	158
5.41.3	lis_log_msgC.c (Source File: lis_log_msgC.c)	159
5.41.4	lis_log_msgC (Source File: lis_log_msgC.c)	159
5.41.5	lis_memory_managementC (Source File: lis_memory_managementC.c)	159
5.41.6	lis_malloc (Source File: lis_memory_managementC.c)	159
5.41.7	lis_realloc (Source File: lis_memory_managementC.c)	159
5.41.8	listask_for_point (Source File: listask_for_point.c)	159

IV Plugin Interfaces in LIS	160
6 Plugin interfaces in LIS	161
6.1 Fortran: Module Interface runmode_pluginMod (Source File: runmode_pluginMod.F90)	161
6.1.1 runmode_plugin (Source File: runmode_pluginMod.F90)	161
6.2 Fortran: Module Interface domain_pluginMod (Source File: domain_pluginMod.F90)	162
6.2.1 domain_plugin (Source File: domain_pluginMod.F90)	162
6.3 Fortran: Module Interface baseforcing_pluginMod (Source File: baseforcing_pluginMod.F90) .	162
6.3.1 baseforcing_plugin (Source File: baseforcing_pluginMod.F90)	163
6.4 Fortran: Module Interface suppforcing_pluginMod (Source File: suppforcing_pluginMod.F90)	163
6.4.1 suppforcing_plugin (Source File: suppforcing_pluginMod.F90)	164
6.5 Fortran: Module Interface lsm_pluginMod (Source File: lsm_pluginMod.F90)	165
6.5.1 lsm_plugin (Source File: lsm_pluginMod.F90)	165
6.6 Fortran: Module Interface param_pluginMod (Source File: param_pluginMod.F90)	167
6.6.1 topo_plugin (Source File: param_pluginMod.F90)	167
6.6.2 soils_plugin (Source File: param_pluginMod.F90)	168
6.6.3 laisai_plugin (Source File: param_pluginMod.F90)	169
6.6.4 landcover_plugin (Source File: param_pluginMod.F90)	170
6.6.5 tbot_plugin (Source File: param_pluginMod.F90)	170
6.6.6 gfrac_plugin (Source File: param_pluginMod.F90)	171
6.6.7 alb_plugin (Source File: param_pluginMod.F90)	171
6.6.8 snow_plugin (Source File: param_pluginMod.F90)	172
6.7 Fortran: Module Interface dataassim_pluginMod.F90 (Source File: dataassim_pluginMod.F90)	173
6.7.1 dataassim_plugin (Source File: dataassim_pluginMod.F90)	173
6.8 Fortran: Module Interface dataobs_pluginMod (Source File: dataobs_pluginMod.F90)	174
6.8.1 dataobs_plugin (Source File: dataobs_pluginMod.F90)	174
6.9 Fortran: Module Interface perturb_pluginMod (Source File: perturb_pluginMod.F90)	175
6.9.1 perturb_plugin (Source File: perturb_pluginMod.F90)	175
V Interpolation Tools in LIS	176
7 Interpolation tools in LIS	177
7.0.2 bilinear_interp_input (Source File: bilinear_interp_input.F90)	177
7.0.3 bilinear_interp (Source File: bilinear_interp.F90)	178
7.0.4 conserv_interp_input (Source File: conserv_interp_input.F90)	179
7.0.5 conserv_interp (Source File: conserv_interp.F90)	180
7.0.6 neighbor_interp_input (Source File: neighbor_interp_input.F90)	182
7.0.7 neighbor_interp (Source File: neighbor_interp.F90)	183
7.0.8 compute_earth_coord (Source File: compute_earth_coord.F90)	184
7.0.9 compute_earth_coord_latlon (Source File: compute_earth_coord_latlon.F90)	186
7.0.10 compute_earth_coord_merc (Source File: compute_earth_coord_merc.F90)	186
7.0.11 compute_earth_coord_lambert (Source File: compute_earth_coord_lambert.F90)	187
7.0.12 compute_earth_coord_gauss (Source File: compute_earth_coord_gauss.F90)	188
7.0.13 compute_earth_coord_polar (Source File: compute_earth_coord_polar.F90)	189
7.0.14 compute_grid_coord (Source File: compute_grid_coord.F90)	190
7.0.15 compute_grid_coord_latlon (Source File: compute_grid_coord_latlon.F90)	191
7.0.16 compute_grid_coord_merc (Source File: compute_grid_coord_merc.F90)	192
7.0.17 compute_grid_coord_lambert (Source File: compute_grid_coord_lambert.F90)	193
7.0.18 compute_grid_coord_gauss (Source File: compute_grid_coord_gauss.F90)	194
7.0.19 compute_grid_coord_polar (Source File: compute_grid_coord_polar.F90)	194

7.0.20	get_field_pos (Source File: get_fieldpos.F90)	196
7.0.21	polfixs (Source File: polfixs.F90)	196
7.0.22	latlonTopolar (Source File: latlonTopolar.F90)	197
7.0.23	polarToLatLon (Source File: polarToLatLon.F90)	198
7.0.24	lltops (Source File: lltops.F90)	199
7.0.25	pstoll (Source File: pstoll.F90)	200
7.1	Fortran: Module Interface map_utils (Source File: map_utils.F90)	200
7.1.1	Supported Projections	201
7.1.2	Remarks	201
7.1.3	Assumptions	201
7.1.4	Data Definitions:	201
7.1.5	Usage	202
7.1.6	map_init (Source File: map_utils.F90)	203
7.1.7	map_set (Source File: map_utils.F90)	203
7.1.8	latlon_to_ij (Source File: map_utils.F90)	204
7.1.9	ij_to_latlon (Source File: map_utils.F90)	205
7.1.10	compute_stnwts (Source File: compute_stnwts.F90)	206
7.1.11	interp_stndata (Source File: interp_stndata.F90)	206
7.1.12	normalize_stnwts (Source File: normalize_stnwts.F90)	207
VI	User-defined extensible components in LIS	208
8	User-defined extensible components in LIS	209
VII	Running Modes in LIS	210
9	Running Modes in LIS	211
10	Retrospective Running Mode	211
10.1	Fortran: Module Interface retrospective_runMod (Source File: retrospective_runMod.F90)	211
10.1.1	lis_init_retrospective (Source File: retrospective_runMod.F90)	211
10.1.2	lis_run_retrospective (Source File: retrospective_runMod.F90)	212
10.1.3	lis_final_retrospective (Source File: retrospective_runMod.F90)	213
11	AGRMET Running Mode	213
11.1	Fortran: Module Interface agrmet_runMod (Source File: agrmet_runMod.F90)	214
11.1.1	lis_init_agrmet (Source File: agrmet_runMod.F90)	214
11.1.2	lis_run_agrmet (Source File: agrmet_runMod.F90)	215
11.1.3	lis_final_agrmet (Source File: agrmet_runMod.F90)	216
VIII	Domain implementations in LIS	217
12	Domain implementations in LIS	218
13	Lat/Lon domain	218
13.0.4	createtiles_latlon (Source File: createtiles_latlon.F90)	218
13.0.5	calculate_domveg (Source File: calculate_domveg.F90)	219
13.0.6	create_vegtilespace_latlon (Source File: create_vegtilespace_latlon.F90)	219
13.0.7	readinput_latlon (Source File: readinput_latlon.F90)	220

14 Mercator domain	221
14.0.8 createtiles_merc (Source File: createtiles_merc.F90)	221
14.0.9 create_vegtilespace_merc (Source File: create_vegtilespace_merc.F90)	222
14.0.10 readinput_merc (Source File: readinput_merc.F90)	222
15 Lambert Conformal domain	223
15.0.11 createtiles_lambert (Source File: createtiles_lambert.F90)	223
15.0.12 create_vegtilespace_lambert (Source File: create_vegtilespace_lambert.F90)	224
15.0.13 readinput_lambert (Source File: readinput_lambert.F90)	225
16 Polar stereographic domain	225
16.0.14 createtiles_polar (Source File: createtiles_polar.F90)	226
16.0.15 create_vegtilespace_polar (Source File: create_vegtilespace_polar.F90)	226
16.0.16 readinput_polar (Source File: readinput_polar.F90)	227
17 GSWP domain	228
17.0.17 createtiles_gswp (Source File: createtiles_gswp.F90)	228
17.0.18 readinput_gswp (Source File: readinput_gswp.F90)	228
18 AFWA/AGRMET domain	229
18.0.19 createtiles_afwa (Source File: createtiles_afwa.F90)	229
18.0.20 readinput_afwa (Source File: readinput_afwa.F90)	229
IX Land Surface Parameters in LIS	231
19 Land Surface Parameters in LIS	232
20 Landmask/Landcover Data	232
20.0.21 read_latlon_umdmask (Source File: read_latlon_umdmask.F90)	232
20.0.22 read_latlon_umdlc (Source File: read_latlon_umdlc.F90)	232
20.0.23 read_latlon_usgsmask (Source File: read_latlon_usgsmask.F90)	233
20.0.24 read_latlon_usgslc (Source File: read_latlon_usgslc.F90)	234
20.0.25 read_lambert_umdmask (Source File: read_lambert_umdmask.F90)	234
20.0.26 read_lambert_umdlc (Source File: read_lambert_umdlc.F90)	235
20.0.27 read_lambert_usgsmask (Source File: read_lambert_usgsmask.F90)	236
20.0.28 read_lambert_usgslc (Source File: read_lambert_usgslc.F90)	236
20.0.29 read_merc_umdmask (Source File: read_merc_umdmask.F90)	237
20.0.30 read_merc_umdlc (Source File: read_merc_umdlc.F90)	238
20.0.31 read_merc_usgsmask (Source File: read_merc_usgsmask.F90)	238
20.0.32 read_merc_usgslc (Source File: read_merc_usgslc.F90)	239
20.0.33 read_polar_umdmask (Source File: read_polar_umdmask.F90)	240
20.0.34 read_polar_umdlc (Source File: read_polar_umdlc.F90)	240
20.0.35 read_polar_usgsmask (Source File: read_polar_usgsmask.F90)	241
20.0.36 read_polar_usgslc (Source File: read_polar_usgslc.F90)	242
20.0.37 read_afwalc (Source File: read_afwalc.F90)	242
20.0.38 read_afwamask (Source File: read_afwamask.F90)	243

21 Albedo Data	243
21.0.39 read_gswpalbedo (Source File: read_gswpalbedo.F90)	243
21.0.40 read_gswpmxsnoalb (Source File: read_gswpmxsnoalb.F90)	244
21.0.41 read_lcalbedo (Source File: read_lcalbedo.F90)	245
21.0.42 read_lcmxsnoalb (Source File: read_lcmxsnoalb.F90)	246
21.0.43 read_llalbedo (Source File: read_llalbedo.F90)	246
21.0.44 read_llmxsnoalb (Source File: read_llmxsnoalb.F90)	247
21.0.45 read_mercalbedo (Source File: read_mercalbedo.F90)	247
21.0.46 read_mercmxsnoalb (Source File: read_mercmxsnoalb.F90)	248
21.0.47 read_psalbedo (Source File: read_psalbedo.F90)	249
21.0.48 read_psmxsnoalb (Source File: read_psmxsnoalb.F90)	249
21.0.49 read_afwamxsnoalb (Source File: read_afwamxsnoalb.F90)	250
22 Greenness Fraction Data	250
22.0.50 read_gswpgfrac (Source File: read_gswpgfrac.F90)	250
22.0.51 read_lcgfrac (Source File: read_lcgfrac.F90)	251
22.0.52 read_llgfrac (Source File: read_llgfrac.F90)	252
22.0.53 read_mercgfrac (Source File: read_mercgfrac.F90)	252
22.0.54 read_psgfrac (Source File: read_psgfrac.F90)	253
22.0.55 read_afwagfrac (Source File: read_afwagfrac.F90)	254
23 Leaf Area Index Data	254
23.0.56 read_gswplai (Source File: read_gswplai.F90)	254
23.0.57 read_gswpsai (Source File: read_gswpsai.F90)	255
23.0.58 read_ll_avhrrlai (Source File: read_ll_avhrrlai.F90)	256
23.0.59 read_ll_avhrrsai (Source File: read_ll_avhrrsai.F90)	256
23.0.60 climatologylai_llread (Source File: climatologylai_llread.F90)	257
23.0.61 climatologysai_llread (Source File: climatologysai_llread.F90)	257
24 Soils	258
24.0.62 read_gswp_bexp (Source File: read_gswp_bexp.F90)	258
24.0.63 read_gswp_ksat (Source File: read_gswp_ksat.F90)	259
24.0.64 read_gswp_psisat (Source File: read_gswp_psisat.F90)	259
24.0.65 read_gswp_soilclass (Source File: read_gswp_soilclass.F90)	260
24.0.66 read_gswpclay (Source File: read_gswpclay.F90)	260
24.0.67 read_gswpsand (Source File: read_gswpsand.F90)	261
24.0.68 read_gswpsilt (Source File: read_gswpsilt.F90)	262
24.0.69 read_lambert_faocl (Source File: read_lambert_faocl.F90)	262
24.0.70 read_lambert_faocolor (Source File: read_lambert_faocolor.F90)	263
24.0.71 read_lambert_faosand (Source File: read_lambert_faosand.F90)	264
24.0.72 read_lambert_faosilt (Source File: read_lambert_faosilt.F90)	264
24.0.73 read_lambert_statsgoclay (Source File: read_lambert_statsgoclay.F90)	265
24.0.74 read_lambert_statgosand (Source File: read_lambert_statgosand.F90)	266
24.0.75 read_lambert_statgosilt (Source File: read_lambert_statgosilt.F90)	266
24.0.76 read_lambert_statgotexture (Source File: read_lambert_statgotexture.F90)	267
24.0.77 read_faocl (Source File: read_faocl.F90)	268
24.0.78 read_faocolor (Source File: read_faocolor.F90)	268
24.0.79 read_faosand (Source File: read_faosand.F90)	269
24.0.80 read_faosilt (Source File: read_faosilt.F90)	269
24.0.81 read_statsgoclay (Source File: read_statsgoclay.F90)	270
24.0.82 read_statgosand (Source File: read_statgosand.F90)	270

24.0.83	read_statsgosilt (Source File: read_statsgosilt.F90)	271
24.0.84	read_statsgotexture (Source File: read_statsgotexture.F90)	271
24.0.85	read_merc_faocl (Source File: read_merc_faocl.F90)	272
24.0.86	read_merc_faocolor (Source File: read_merc_faocolor.F90)	272
24.0.87	read_merc_faosand (Source File: read_merc_faosand.F90)	273
24.0.88	read_merc_faosilt (Source File: read_merc_faosilt.F90)	274
24.0.89	read_merc_statsgoclay (Source File: read_merc_statsgoclay.F90)	274
24.0.90	read_merc_statsgosand (Source File: read_merc_statsgosand.F90)	275
24.0.91	read_merc_statsgosilt (Source File: read_merc_statsgosilt.F90)	276
24.0.92	read_merc_statsgotexture (Source File: read_merc_statsgotexture.F90)	276
24.0.93	read_polar_faocl (Source File: read_polar_faocl.F90)	277
24.0.94	read_polar_faocolor (Source File: read_polar_faocolor.F90)	278
24.0.95	read_polar_faosand (Source File: read_polar_faosand.F90)	278
24.0.96	read_polar_faosilt (Source File: read_polar_faosilt.F90)	279
24.0.97	read_polar_statsgoclay (Source File: read_polar_statsgoclay.F90)	280
24.0.98	read_polar_statsgosand (Source File: read_polar_statsgosand.F90)	280
24.0.99	read_polar_statsgosilt (Source File: read_polar_statsgosilt.F90)	281
24.0.100	read_polar_statsgotexture (Source File: read_polar_statsgotexture.F90)	282
24.0.101	read_afwa_soils (Source File: read_afwa_soils.F90)	282
25 Snow		283
25.0.102	read_llsnowdepth (Source File: read_llsnowdepth.F90)	283
25.0.103	read_lcsnowdepth (Source File: read_lcsnowdepth.F90)	283
25.0.104	read_mercsnowdepth (Source File: read_mercsnowdepth.F90)	284
25.0.105	read_polarsnowdepth (Source File: read_polarsnowdepth.F90)	284
26 Topography		285
26.0.106	read_lambert_elev_gtopo30 (Source File: read_lambert_elev_gtopo30.F90)	285
26.0.107	read_elev_gtopo30 (Source File: read_elev_gtopo30.F90)	285
26.0.108	read_slope_gtopo30 (Source File: read_slope_gtopo30.F90)	286
26.0.109	read_aspect_gtopo30 (Source File: read_aspect_gtopo30.F90)	287
26.0.110	read_curv_gtopo30 (Source File: read_curv_gtopo30.F90)	287
26.0.111	read_merc_elev_gtopo30 (Source File: read_merc_elev_gtopo30.F90)	288
26.0.112	read_polar_elev_gtopo30 (Source File: read_polar_elev_gtopo30.F90)	288
27 Bottom Temperature		289
27.0.113	read_gswptbot (Source File: read_gswptbot.F90)	289
27.0.114	read_lambert_statictbot (Source File: read_lambert_statictbot.F90)	290
27.0.115	read_statictbot (Source File: read_statictbot.F90)	290
27.0.116	read_merc_statictbot (Source File: read_merc_statictbot.F90)	291
27.0.117	read_polar_statictbot (Source File: read_polar_statictbot.F90)	291
27.0.118	read_AFWAtbot (Source File: read_afwatbot.F90)	292
X Base forcing analyses in LIS		293
28 Overview		294

29 AGRMET	294
29.1 Fortran: Module Interface agrmetdomain_module (Source File: agrmetdomain_module.F90)	294
29.1.1 defineNativeAGRMET (Source File: agrmetdomain_module.F90)	298
29.1.2 readagrmetcrd (Source File: readagrmetcrd.F90)	299
29.1.3 read_agrmetmask (Source File: read_agrmetmask.F90)	300
29.1.4 get_agrmetmask_filename (Source File: read_agrmetmask.F90)	300
29.1.5 read_agrmetterrain (Source File: read_agrmetterrain.F90)	301
29.1.6 get_agrmetterrain_filename (Source File: read_agrmetterrain.F90)	302
29.1.7 read_pcpcntm (Source File: read_pcpcntm.F90)	302
29.1.8 get_agrmetpcpcntm_filename (Source File: read_pcpcntm.F90)	303
29.1.9 read_sfcalcncntm (Source File: read_sfcalcncntm.F90)	303
29.1.10 get_agrmetsfcncntm_filename (Source File: read_sfcalcncntm.F90)	304
29.1.11 read_pcpclimodata (Source File: read_pcpclimodata.F90)	305
29.1.12 getclimortnfilename (Source File: read_pcpclimodata.F90)	306
29.1.13 getclimoprcfilename (Source File: read_pcpclimodata.F90)	307
29.1.14 getclimoppdfilename (Source File: read_pcpclimodata.F90)	307
29.1.15 getcli (Source File: getcli.F90)	308
29.1.16 getagrmet (Source File: getagrmet.F90)	309
29.1.17 retrieve_agrmetvar (Source File: retrieve_agrmetvar.F90)	310
29.1.18 readagrmetforcing (Source File: readagrmetforcing.F90)	310
29.1.19 agrmet_cdfs_pcts_filename (Source File: readagrmetforcing.F90)	313
29.1.20 agrmet_cdfs_type_filename (Source File: readagrmetforcing.F90)	314
29.1.21 agrmet_cdfs_hgts_filename (Source File: readagrmetforcing.F90)	314
29.1.22 agrmet_cdfs_pixltime_filename (Source File: readagrmetforcing.F90)	315
29.1.23 find_agrsfc_dataindex (Source File: readagrmetforcing.F90)	316
29.1.24 readagrmetforcinganalysis (Source File: readagrmetforcinganalysis.F90)	316
29.1.25 agrmet_sfctmp_filename (Source File: readagrmetforcinganalysis.F90)	319
29.1.26 agrmet_sfcrlh_filename (Source File: readagrmetforcinganalysis.F90)	320
29.1.27 agrmet_sfcpers_filename (Source File: readagrmetforcinganalysis.F90)	320
29.1.28 agrmet_sfcpd_filename (Source File: readagrmetforcinganalysis.F90)	321
29.1.29 agrmet_sfcalc (Source File: agrmet_sfcalc.F90)	322
29.1.30 find_agrflld_starttime (Source File: agrmet_sfcalc.F90)	324
29.1.31 fldbld (Source File: fldbld.F90)	324
29.1.32 getAVNfilename (Source File: fldbld.F90)	326
29.1.33 (Source File: fldbld.F90)	326
29.1.34 getDataLevelsFilename (Source File: fldbld.F90)	326
29.1.35 fldbld_setup (Source File: fldbld_setup.F90)	327
29.1.36 fldbld_read (Source File: fldbld_read.F90)	328
29.1.37 calcrh (Source File: calcrh.F90)	330
29.1.38 Method	331
29.1.39 fg_to_agr (Source File: fg_to_agr.F90)	331
29.1.40 fgfill (Source File: fgfill.F90)	333
29.1.41 sfval (Source File: sfval.F90)	335
29.1.42 getsfc (Source File: getsfc.F90)	338
29.1.43 getsfcobsfilename (Source File: getsfc.F90)	340
29.1.44 sfcalc_barnes (Source File: sfcalc_barnes.F90)	341
29.1.45 interp_agrmetvar (Source File: interp_agrmetvar.F90)	343
29.1.46 agrmet_fillgaps (Source File: agrmet_fillgaps.F90)	343
29.1.47 agrmet_svp (Source File: agrmet_svp.F90)	344
29.1.48 agrmet_longwv (Source File: agrmet_longwv.F90)	345
29.1.49 agrmet_calc_albedo (Source File: agrmet_calc_albedo.F90)	346

29.1.50 agrmet_tr_coeffs (Source File: agrmet_tr_coeffs.F90)	347
29.1.51 agrmet_solar (Source File: agrmet_solar.F90)	349
29.1.52 agrmet_tpcnv (Source File: agrmet_tpcnv.F90)	351
29.1.53 agrmet_trnref (Source File: agrmet_trnref.F90)	352
29.1.54 agrmet_bakfac (Source File: agrmet_bakfac.F90)	353
29.1.55 agrmet_w (Source File: agrmet_w.F90)	354
29.1.56 agrmet_trpoly (Source File: agrmet_trpoly.F90)	355
29.1.57 agrmet_trcalc (Source File: agrmet_trcalc.F90)	356
29.1.58 loadcloud (Source File: loadcloud.F90)	358
29.1.59 agrmet_bndslc (Source File: agrmet_bndslc.F90)	360
29.1.60 readagrmetpcpforcing (Source File: readagrmetpcpforcing.F90)	361
29.1.61 find_agrppcp_starttime (Source File: readagrmetpcpforcing.F90)	363
29.1.62 find_agrppcp_readtime (Source File: readagrmetpcpforcing.F90)	364
29.1.63 readagrmetpcpforcinganalysis (Source File: readagrmetpcpforcinganalysis.F90)	364
29.1.64 get_agrppcp_readtime (Source File: readagrmetpcpforcinganalysis.F90)	366
29.1.65 getpcpobs (Source File: getpcpobs.F90)	366
29.1.66 agrmetpcpobsfilename (Source File: getpcpobs.F90)	369
29.1.67 setpathpcpobs (Source File: getpcpobs.F90)	369
29.1.68 getpwe (Source File: getpwe.F90)	370
29.1.69 makpwe (Source File: makpwe.F90)	372
29.1.70 storeobs (Source File: storeobs.F90)	373
29.1.71 processobs (Source File: processobs.F90)	374
29.1.72 pcpobs_search (Source File: pcpobs_search.F90)	378
29.1.73 makest (Source File: makest.F90)	378
29.1.74 cliest (Source File: cliest.F90)	382
29.1.75 smiest (Source File: smiest.F90)	383
29.1.76 agrmet_ssmiprec_filename (Source File: smiest.F90)	384
29.1.77 cdfs2_est (Source File: cdfs2_est.F90)	385
29.1.78 agrmet_cdfs_totalcld_filename (Source File: cdfs2_est.F90)	388
29.1.79 geoest (Source File: geoest.F90)	389
29.1.80 agrmet_geoprec_filename (Source File: geoest.F90)	391
29.1.81 agrmet_georank_filename (Source File: geoest.F90)	392
29.1.82 calest (Source File: calest.F90)	392
29.1.83 phsrel12 (Source File: phsrel12.F90)	395
29.1.84 parse12 (Source File: parse12.F90)	397
29.1.85 phsrel6 (Source File: phsrel6.F90)	398
29.1.86 parse6 (Source File: parse6.F90)	399
29.1.87 agrmet_valid (Source File: agrmet_valid.F90)	400
29.1.88 pcp_barnes (Source File: pcp_barnes.F90)	402
29.1.89 make03 (Source File: make03.F90)	405
29.1.90 julhr_date10 (Source File: julhr_date10.F90)	406
29.1.91 time_interp_agrmet (Source File: time_interp_agrmet.F90)	407
29.1.92 agrmetforcing_finalize (Source File: agrmetforcing_finalize.F90)	407

30 BERG	408
30.1 Fortran: Module Interface bergdomain_module (Source File: bergdomain_module.F90)	408
30.1.1 defineNativeBERG (Source File: bergdomain_module.F90)	409
30.1.2 readbergcrd (Source File: readbergcrd.F90)	410
30.1.3 read_berg_elev (Source File: read_berg_elev.F90)	410
30.1.4 readbergmask (Source File: readbergmask.F90)	410
30.1.5 getberg (Source File: getberg.F90)	411

30.1.6	retberg (Source File: retberg.F90)	412
30.1.7	berggrid_2_lisgrid (Source File: retberg.F90)	413
30.1.8	fillgaps (Source File: retberg.F90)	414
30.1.9	time_interp_berg (Source File: time_interp_berg.F90)	414
30.1.10	geogfill2 (Source File: geogfill2.F90)	415
30.1.11	bergforcing_finalize (Source File: bergforcing_finalize.F90)	416
31	ECMWF	416
31.1	Fortran: Module Interface ecmwfdomain_module (Source File: ecmwfdomain_module.F90)	416
31.1.1	defineNativeECMWF (Source File: ecmwfdomain_module.F90)	418
31.1.2	readecmwfcrd (Source File: readecmwfcrd.F90)	418
31.1.3	getecmwf (Source File: getecmwf.F90)	419
31.1.4	retecwmwf (Source File: retecwmwf.F90)	419
31.1.5	ret_inst3 (Source File: retecwmwf.F90)	421
31.1.6	ret_accum (Source File: retecwmwf.F90)	421
31.1.7	interp_iv (Source File: retecwmwf.F90)	422
31.1.8	time_interp_ecmwf (Source File: time_interp_ecmwf.F90)	423
31.1.9	ecmwfforcing_finalize (Source File: ecmwfforcing_finalize.F90)	424
32	GDAS	425
32.1	Fortran: Module Interface gdasdomain_module (Source File: gdasdomain_module.F90)	425
32.1.1	defineNativeGDAS (Source File: gdasdomain_module.F90)	426
32.1.2	readgdascrd (Source File: readgdascrd.F90)	427
32.1.3	read_gdas_elev (Source File: read_gdas_elev.F90)	427
32.1.4	getgdas (Source File: getgdas.F90)	428
32.1.5	gdasfile (Source File: getgdas.F90)	429
32.1.6	gdasfilef06 (Source File: getgdas.F90)	430
32.1.7	retgdas (Source File: retgdas.F90)	430
32.1.8	interp_gdas (Source File: retgdas.F90)	431
32.1.9	time_interp_gdas (Source File: time_interp_gdas.F90)	432
32.2	Fortran: Module Interface gdasforcing_finalize (Source File: gdasforcing_finalize.F90)	433
33	GEOS	434
33.1	Fortran: Module Interface geosdomain_module (Source File: geosdomain_module.F90)	434
33.1.1	defineNativeGEOS (Source File: geosdomain_module.F90)	435
33.1.2	readgeoscrd (Source File: readgeoscrd.F90)	436
33.1.3	getgeos (Source File: getgeos.F90)	436
33.1.4	geosfile (Source File: getgeos.F90)	437
33.1.5	readgeos (Source File: readgeos.F90)	438
33.1.6	time_interp_geos (Source File: time_interp_geos.F90)	439
33.1.7	geosforcing_finalize (Source File: geosforcing_finalize.F90)	440
34	GSWP	440
34.1	Fortran: Module Interface gswpdomain_module (Source File: gswpdomain_module.F90)	440
34.1.1	defineNativeGSWP (Source File: gswpdomain_module.F90)	441
34.1.2	readgswpcrd (Source File: readgswpcrd.F90)	442
34.1.3	getgswp (Source File: getgswp.F90)	442
34.1.4	readgswp (Source File: readgswp.F90)	443
34.1.5	time_interp_gswp (Source File: time_interp_gswp.F90)	444
34.1.6	finterp (Source File: finterp.F90)	444
34.2	Fortran: Module Interface gswpforcing_finalize (Source File: gswpforcing_finalize.F90)	445

35 NLDAS	446
35.1 Fortran: Module Interface nldasdomain_module (Source File: nldasdomain_module.F90)	446
35.1.1 defineNativeNLDAS (Source File: nldasdomain_module.F90)	447
35.1.2 readnldascrd (Source File: readnldascrd.F90)	448
35.1.3 read_nldas_elev (Source File: read_nldas_elev.F90)	448
35.1.4 getnldas (Source File: getnldas.F90)	449
35.1.5 nldasfile (Source File: getnldas.F90)	450
35.1.6 retrnldas (Source File: retrnldas.F90)	451
35.1.7 interp_nldas (Source File: retrnldas.F90)	452
35.1.8 time_interp_nldas (Source File: time_interp_nldas.F90)	453
35.2 Fortran: Module Interface nldasforcing_finalize (Source File: nldasforcing_finalize.F90)	454
XI Supplemental forcing analyses in LIS	455
36 Overview	456
37 CMAP	457
37.1 Fortran: Module Interface cmapdomain_module (Source File: cmapdomain_module.F90)	457
37.1.1 defineNativeCMAP (Source File: cmapdomain_module.F90)	458
37.1.2 readcmapcrd (Source File: readcmapcrd.F90)	458
37.1.3 getcmap (Source File: getcmap.F90)	459
37.1.4 cmapfile (Source File: getcmap.F90)	459
37.1.5 glbprecip_cmap (Source File: glbprecip_cmap.F90)	460
37.1.6 interp_cmap (Source File: interp_cmap.F90)	461
37.1.7 time_interp_cmap (Source File: time_interp_cmap.F90)	461
38 Huffman	462
38.1 Fortran: Module Interface huffdomain_module (Source File: huffdomain_module.F90)	462
38.1.1 defineNativeHuff (Source File: huffdomain_module.F90)	463
38.1.2 readhuffcrd (Source File: readhuffcrd.F90)	464
38.1.3 gethuff (Source File: gethuff.F90)	464
38.1.4 hufffile (Source File: gethuff.F90)	465
38.1.5 glbprecip_huff (Source File: glbprecip_huff.F90)	466
38.1.6 interp_huff (Source File: interp_huff.F90)	466
38.1.7 time_interp_huff (Source File: time_interp_huff.F90)	467
39 CMORPH	468
39.1 Fortran: Module Interface cmordomain_module (Source File: cmordomain_module.F90)	468
39.1.1 defineNativeCMORPH (Source File: cmordomain_module.F90)	469
39.1.2 readcmorcrd (Source File: readcmorcrd.F90)	469
39.1.3 getcmor (Source File: getcmor.F90)	470
39.1.4 cmorfile (Source File: getcmor.F90)	471
39.1.5 glbprecip_cmor (Source File: glbprecip_cmor.F90)	471
39.1.6 interp_cmor (Source File: interp_cmor.F90)	472
39.1.7 time_interp_cmor (Source File: time_interp_cmor.F90)	473
XII Land Surface Models in LIS	474
40 Land Surface Models in LIS	475

41 Noah land surface model version 2.6	475
41.1 Fortran: Module Interface noah_module (Source File: noah_module.F90)	475
41.2 Fortran: Module Interface noah_varder (Source File: noah_varder.F90)	477
41.2.1 noah_varder_ini (Source File: noah_varder.F90)	478
41.2.2 readnoahcrd (Source File: readnoahcrd.F90)	479
41.2.3 noah_setvegparms (Source File: noah_setvegparms.F90)	479
41.2.4 noah_setup (Source File: noah_setup.F90)	480
41.2.5 noah_setsoils (Source File: noah_setsoils.F90)	480
41.2.6 noah_settbot (Source File: noah_settbot.F90)	481
41.2.7 noah_coldstart (Source File: noah_coldstart.F90)	482
41.2.8 noah_dynsetup (Source File: noah_dynsetup.F90)	482
41.2.9 noah_f2t (Source File: noah_f2t.F90)	482
41.2.10 noah_main (Source File: noah_main.F90)	483
41.2.11 noah_output (Source File: noah_output.F90)	483
41.2.12 noah_binout (Source File: noah_binout.F90)	484
41.2.13 noah_gribout (Source File: noah_gribout.F90)	485
41.2.14 noah_ncdfout (Source File: noah_ncdfout.F90)	486
41.2.15 noah_totinit (Source File: noah_totinit.F90)	487
41.2.16 noah_writerst (Source File: noah_writerst.F90)	487
41.2.17 noah_dump_restart (Source File: noah_writerst.F90)	488
41.2.18 noahrst (Source File: noahrst.F90)	489
41.2.19 noah_finalize (Source File: noah_finalize.F90)	490
42 Community Land Model version 2.0	490
42.1 Fortran: Module Interface clmtype (Source File: clmtype.F90)	490
42.2 Fortran: Module Interface clm_varder (Source File: clm_varder.F90)	495
42.2.1 clm_varder_ini (Source File: clm_varder.F90)	496
42.2.2 clm_varder_init (Source File: clm_varder.F90)	497
42.3 Fortran: Module Interface atmdrvMod (Source File: atmdrvMod.F90)	497
42.3.1 atmdrv (Source File: atmdrvMod.F90)	497
42.3.2 clm2_setup (Source File: clm2_setup.F90)	499
42.3.3 iniTimeConst (Source File: iniTimeConst.F90)	499
42.3.4 iniTimeVar (Source File: iniTimeVar.F90)	500
42.3.5 clm2_dynsetup (Source File: clm2_dynsetup.F90)	501
42.3.6 clm2_main (Source File: clm2_main.F90)	502
42.3.7 clm2_output (Source File: clm2_output.F90)	503
42.3.8 clm2_binout (Source File: clm2_binout.F90)	504
42.3.9 clm2_gribout (Source File: clm2_gribout.F90)	505
42.3.10 clm2_totinit (Source File: clm2_totinit.F90)	505
42.3.11 clm2wrst (Source File: clm2wrst.F90)	506
42.3.12 clm_dump_restart (Source File: clm2wrst.F90)	506
42.3.13 clm2_restart (Source File: clm2_restart.F90)	507
42.3.14 clm2_finalize (Source File: clm2_finalize.F90)	508

Part I

LIS Overview

1 Release Notes

This is the reference manual for Land Information System version 4.2. In this version, a number of usable features have been added to the LIS code, including a scalable, model-independent parallel programming support, highly interactive and user-friendly configuration management, streamlined diagnostic and error logging, and extensible plugin interfaces specified for different running modes, land surface models, parameters, domains, data assimilation, and various forcing analyses.

2 What is the Land Information System?

The Land Information System (LIS;[14, 10]) is a Land Data Assimilation System (LDAS) that unifies and extends the capabilities of the 1/4 degree Global LDAS (GLDAS; [16]) and the 1/8th degree North American LDAS (NLDAS; [11]) to determine water and energy states (e.g. soil moisture, snow) and fluxes (e.g. evaporation, transpiration, runoff) at 1km and other spatial resolutions, and at 1-hour and finer temporal resolutions. The 1km capability of LIS allows it to take advantage of the latest EOS-era observations, such as the MODIS leaf area index, snow cover and surface temperature, at their full resolution. LIS features a high performance and flexible design, provides infrastructure for data integration and assimilation, and operates primarily on an ensemble of land surface models for extension over user-specified regional or global domains. The LIS framework is designed using advanced software engineering principles to enable the reuse and community sharing of modeling tools, data resources, and assimilation algorithms. LIS provides generic, model-independent support for high performance computing, resource management, data handling, inter-language support and other functions. The LIS software framework is designed as an object-oriented framework, with explicit abstract interfaces defined for customization and extension for different applications. These extensible interfaces allow the incorporation of new domains, land surface models, land surface parameter data, meteorological analyses, and data assimilation tools into LIS. As these interfaces are designed to remain independent from specific models, the component-style specification of the system allows rapid prototyping and development of new applications.

3 The LIS Reference Manual

This manual provides a listing of the LIS modules, interfaces and methods. This section is followed by the description of the offline driver program for LIS. Parts III and IV describe the core structures in LIS and the extensible plugin interfaces, respectively. LIS provides a generic spatial interpolation tool to support the geospatial transformation of many input data sets. This package is described in Part V.

As described above, a main advantage of LIS is the ability to include new, user-defined components. Implementation of some such components are described in parts VI to XII. These parts describe implementations of various running modes (e.g. retrospective, forecast), domains (e.g., lat/lon, GSWP, polar stereographic), land surface parameters (e.g., soils, LAI), forcing analyses (e.g., AGRMET, ECMWF, CMORPH), and land surface models (e.g., Noah, CLM)

To get started with LIS, please see the LIS user's guide. This document includes installation instructions, description of a lis configuration, and other useful information.

Part II

LIS offline program

4 LIS offline program

4.0.1 lisdrv (Source File: lisdrv.F90)

Main program for LIS in an offline simulation

Main driver program for LIS. It performs four main functions

LIS_config calls the routines to read the runtime configurations

LIS_Init calls the initializations based on the runmode

LIS_run calls the run routines based on the runmode

LIS_finalize calls the cleanup routines

The routines invoked are :

lisconfig (5.1.2)

call to initialize configuration tool and read model independent options

lisinit (5.36.2)

call to initialize lis based on the runmode

lisrun (5.36.4)

call to run lis based on the runmode

lisfinalize (5.36.6)

call to cleanup lis structures based on the runmode

REVISION HISTORY:

```
14Nov02    Sujay Kumar  Initial Specification
21Oct05    Sujay Kumar  Modified to include the runmodes. Switched
                  to a init,run,finalize mode
program lisdrv
```

USES:

```
use lisdrv_module

call lisconfig
call lisinit(lis%runmode)
call lisrun(lis%runmode)
call lisfinalize(lis%runmode)
```

Part III

LIS core structures and methods

5 LIS core structures and methods

LIS core, the central part of the software, is the infrastructure that integrates the use of different components in LIS. The functions performed by LIS core include operations related to the overall control, runtime statistics, inter-language support, error logging and dynamic memory management functions. Routines to manage domain decomposition, load balancing, fault tolerance, etc. are also encapsulated as generic routines in the high performance computing and communications (HPCC) component. The time management tools in LIS provides useful functions for time and data calculations and higher level functions to control model timestepping and alarms. Another tool implemented in the LIS core structure is the generic I/O tool, which provides capabilities to read the input data locally, handling different data formats. also provide support for distributed data output and multiple formats. Other miscellaneous tools incorporated in the LIS core include methods to perform spatial and temporal interpolation, reprojection, domain subsetting etc. The abstractions providing representations for the behavior of LSMs, domains, and data, runmodes, and data assimilation are also incorporated in the LIS core.

5.1 Fortran: Module Interface lisdrv_module (Source File: lisdrv_module.F90)

The code in this file contains the basic datastructures and control routines for the operation of LIS

5.1.1 Overview

This module contains the defintion and specification of the basic datastructures that define a LIS instance. It consists of:

lis datastructure containg generic LIS variables

lisdom datastructure containing grid, tile spaces, grid projection, domain decomposition information.

metadata_output datastructure defining the metadata for model output.

config_lis instance of the configuration class.

REVISION HISTORY:

14 Nov 2002 Sujay Kumar Initial Specification

USES:

```
use lis_module
use grid_module
use tile_module
use listime_mngr
use spmdMod
use output_metaMod
use map_utils
use LIS_ConfigMod

implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: lisconfig
public :: LIS_ticktime
public :: LIS_endofrun
```

```
public :: LIS_TimeToRunNest
public :: LIS_finalize
```

PUBLIC TYPES:

```
public :: lis
public :: lisdom
public :: metadata_output
public :: config_lis
public :: initialized
```

5.1.2 lisconfig (Source File: lisdrv_module.F90)

INTERFACE:

```
interface lisconfig
```

PRIVATE MEMBER FUNCTIONS:

```
module procedure lisconfig_offline
module procedure lisconfig_coupled
```

DESCRIPTION:

This interface provides routines for the setup of LIS configuration management. It initializes the LIS Config utility, reads the model independent specifications, and initializes the SPMD parallel processing mode. This routine also initializes the LIS time manager, and finally allocates memory for the LIS generic datastructures.

5.1.3 lisconfig_offline (Source File: lisdrv_module.F90)

INTERFACE:

```
Private name: call using lisconfig()
subroutine lisconfig_offline()
```

DESCRIPTION:

Performs the initialization of LIS runtime configuration for an offline (uncoupled simulation).
The Calling sequence is :

spmd_init (5.5.1)

 performs SPMD initializations

readcard (5.26.1)

 reads the model independent options

spmd_setup (5.5.4)

 allocates memory for SPMD variables

timemgr_init (5.4.2)

 initializes the time manager

5.1.4 lisconfig_couple (Source File: lisdrv_module.F90)

INTERFACE:

```
Private name: call using lisconfig()
subroutine lisconfig_couple(nx,ny)

implicit none
```

ARGUMENTS:

```
integer,intent(in) :: nx, ny
```

DESCRIPTION:

Performs the initialization of LIS runtime configuration for an coupled simulation. The processor layout is obtained from a parent component.

The arguments are:

nx Number of processors in the East-West direction in a processor layout

ny Number of processors in the North-South direction in a processor layout

The Calling sequence is :

spmd_init (5.5.1)
performs SPMD initializations

readcard (5.26.1)
reads the model independent options

spmd_setup (5.5.4)
allocates memory for SPMD variables

timemgr_init (5.4.2)
initializes the time manager

5.1.5 LIS_ticktime (Source File: lisdrv_module.F90)

INTERFACE:

```
subroutine LIS_ticktime()
```

DESCRIPTION:

This routine calls the time manager to increment the runtime clock by the model timestep.
The Calling sequence is :

advance_timestep (5.4.5)
advances the clock

5.1.6 LIS_endofrun (Source File: lisdrv_module.F90)

INTERFACE:

```
function LIS_endofrun() result(finish)
```

ARGUMENTS:

```
logical :: finish
```

DESCRIPTION:

This function checks to see if the runtime clock has reached the specified stop time of the simulation.
The arguments are:

finish boolean value indicating if the end of simulation is reached.

The calling sequence is:

is_last_step (5.4.11)

```
check if the clock has reached the stop time
```

5.1.7 LIS_timeToRunNest (Source File: lisdrv_module.F90)

INTERFACE:

```
function LIS_timeToRunNest(n) result(check)
```

ARGUMENTS:

```
integer, intent(in) :: n  
logical :: check
```

DESCRIPTION:

Check to see if it is time to run the nest based on the model timestep for that particular domain

n index of the nest or domain

check boolean value indicating if the time to run the nest has reached or not

5.1.8 LIS_finalize (Source File: lisdrv_module.F90)

INTERFACE:

```
subroutine LIS_finalize
```

USES:

```
use spmdMod, only : spmd_finalize
```

DESCRIPTION:

This routine issues the invocation to deallocate and cleanup any allocated data structures.
The calling sequence is:

spmd_finalize (5.5.5)

```
cleanup SPMD structures
```

5.2 Fortran: Module Interface lis_module (Source File: lis_module.F90)

Module for specifying model independent variables in LIS. The module does not contain any variables that are specific to the extensible components in LIS. The module specifies variables for overall run control, options for the choice of parameter datasets, data assimilation choices, and the variables controlling the time management in LIS.

The variables specified in this module include:

runmode choice of running mode in LIS (1-retrospective)

nnest number of subdomains or subnests (0 or higher)

nch This array stores the size of tilespace of the running domain, for each processor, for each nest

glbnch This array stores the size of the tilespace of the overall running domain for each nest

ngrid This array stores the size of gridspace of the running domain, for each processor, for each nest

glbngrid This array stores the size of the gridspace of the overall running domain for each nest

gnc Array containing the East-West grid dimension of the overall running grid for each nest

gnr Array containing the North-South grid dimension of the overall running grid for each nest

lnc Array containing the East-West grid dimension of the running grid for each processor, for each nest

lnr Array containing the North-South grid dimension of the running grid for each processor, for each nest

pnc Array containing the East-West grid dimension of the parameter space for each nest, from which data is subsetted.

pnr Array containing the North-South grid dimension of the parameter space for each nest, from which data is subsetted.

domain Choice of "LIS domain" (Please refer to `domain_module` for the definition of a domain in LIS.)

lsm Choice of the land surface model in LIS.

texturemap boolean value to denote if soil texture map should be used (0- do not use, 1-use). If this value is set to 1, the sand, clay and silt fraction maps will not be read. Otherwise the texture data is read the fraction data will not be used.

vegsrc Choice of the source of landcover data source

soilsrc Choice of the source of soil parameter data source

colorsdc Choice of the soil color data source(0-do not use)

toposrc Choice of the topography data source (0-do not use)

albedosrc Choice of the albedo data source (0-do not use)

gfracsdc Choice of the greenness data source (0-do not use)

porositysrc Choice of the porosity data source (0-do not use)

psisatsrc Choice of the saturated matric potential data source (0-do not use)

ksatsrc Choice of the saturated hydraulic conductivity data source (0-do not use)

bexpsrc Choice of the b parameter data source (0-do not use)

quartzsrc Choice of the quartz data source (0-do not use)

laisrc Choice of the LAI data source (0-do not use)

snowsrc Choice of the snowdepth data source (0-do not use)

maxt Maximum number of tiles per grid to be considered for subgrid tiling

mina Minimum cutoff percentage of vegetation distribution for subgrid tiling

npesx Number of processors in the East-West dimensions for processor layout

npesy Number of processors in the North-South dimensions for processor layout

udef Value to be used as undefined variable

gridDesc Array describing the running grid, for each nest

soil_gridDesc Array describing the soils data grid, for each nest

topo_gridDesc Array describing the topography data grid, for each nest

lc_gridDesc Array describing the landcover data grid, for each nest

forcing Choice of forcing data

ecor Choice of topographical downscaling method

nf Number of forcing variables

rstflag boolean array denoting if the current simulation is a restart or not, for each nest

gridchange boolean array denoting if the native grid for the meteorological forcing need to be changed.

interp Spatial interpolation option (1-bilinear, 2-conservative, 3-neighbor)

tinterp Temporal interpolation option (1-next, 2-uber-next)

shortflag Shortwave radiation source flag (1-instantaneous, 2-time averaged)

longflag Longwave radiation source flag (1-instantaneous, 2-time averaged)

findtime1, findtime2 boolean flags to indicate which time is to be read for temporal interpolation, for each nest

F00_flag,F06_flag flags used to control the forecast forcing read intervals

suppforc Choice of supplemental forcing

perturb Choice of perturbation algorithm (0-do not use)

nt Number of vegetation classes in the landcover dataset

bareclass Index of bare class in the landcover dataset

urbanclass Index of urban class in the landcover dataset

snowclass Index of snow class in the landcover dataset

waterclass Index of water class in the landcover dataset

laiflag flag to control the LAI data read

saiflag flag to control the SAI data read

mfile Name of the landmask file, for each nest

vfile Name of the landcover file, for each nest
safile Name of the sand fraction data file, for each nest
clfle Name of the clay fraction data file, for each nest
sifile Name of the silt fraction data file, for each nest
txtfile Name of the soil texture data file, for each nest
po1file Name of the top layer porosity data file, for each nest
po2file Name of the middle layer porosity data file, for each nest
po3file Name of the bottom layer porosity data file, for each nest
psisatfile Name of the saturated matric potential data file, for each nest
ksatfile Name of the saturated hydraulic conductivity data file, for each nest
bexpfile Name of the b parameter data file, for each nest
qzfile Name of the quartz data file, for each nest
iscfile Name of the soil color data file, for each nest
elevfile Name of the elevation data file, for each nest
slfile Name of the slope data file, for each nest
aspfile Name of the aspect data file, for each nest
curvfile Name of the curvature data file, for each nest
ladir Name of the LAI data source, for each nest
snowdir Name of the snow depth data source, for each nest
gfracfile Name of the greenness data file, for each nest
albfile Name of the climatology albedo data file, for each nest
mxsnal Name of the max snow albedo data file, for each nest
tbotfile Name of the bottom temperature data file, for each nest
albInterval Frequency of albedo climatology in months
gfracInterval Frequency of greenness climatology in months
laitime Variable to keep track of LAI data interval
saitime Variable to keep track of SAI data interval
wfor Option to output forcing data
wopt Output methodology (0=no output, 1-tiled, 2-gridded)
wout Output data format (1-binary, 2-grib, 3-netcdf)
wparam Output parameters (0-do not write, 1-write)
wsingle Option to write each variable to a separate file
expcode Three character experiment code

startcode Start mode (1-restart, 2-coldstart)
plevel Logging level
odir Output directory
dfile Diagnostic output file
sdo_y Starting julian day
sss Starting second
smn Starting minute
shr Starting hour
sda Starting day
smo Starting month
syr Starting year
endcode End mode (1-specific date)
edoy Ending julian day
ess Ending second
emn Ending minute
ehr Ending hour
eda Ending day
emo Ending month
eyr Ending year
endtime Flag to indicate the end of simulation (1-end of simulation)
etime Ending time
egmt Ending time in GMT
nts Array containing the model timestep for each nest
ts Timestep for the clock (minimum timestep of different nests)
tscount Timestep count for each nest
doy Current julian day
ss Current second
mn Current minute
hr Current hour
da Current day
mo Current month
yr Current year
time Current time

gmt Current time in GMT
daalg Choice of data assimilation algorithm
daobs Choice of observation data for assimilation
davar Choice of variable to be assimilated
nensem Number of ensembles per grid

REVISION HISTORY:

15 Oct 1999: Paul Houser; Initial code
12 Apr 2001: Urszula Jambor; Added domain,lsm,& force namefile paramters
30 Jul 2001: Matt Rodell; Add new soil parameter variables
14 Nov 2002; Sujay Kumar; Optimized version for LIS
14 Oct 2003; Sujay Kumar; Removed LSM specific variables.

5.3 Fortran: Module Interface grid_module (Source File: grid_module.F90)

The code in this file provides a description of the grid data structure in LIS

5.3.1 Overview

This module contains the grid data structure used in LIS. The data structure contains the variables specified for a single grid cell. It includes:

lat latitude of the grid cell
lon longitude of the grid cell
forcing An array of meteorological forcing variables
elev Topological elevation of the grid cell
slope Topological slope of the grid cell
aspect Topological aspect of the grid cell
curv Topological curvature of the grid cell

The order of variables in the forcing array is specified in the `baseforcing_module`.

REVISION HISTORY:

15 Oct 1999: Paul Houser; Initial code
15 Oct 2001: Jesse Meng; Revised doc block with forcing array definition
14 Nov 2002: Sujay Kumar; Optimized version of grid_module

5.4 Fortran: Module Interface listime_mngr (Source File: listime_mngr.F90)

This module contains routines for time management. The module provides routines for clock initialization, model timestepping, some basic alarm functions, and other useful time management utilities.

USES:

```
use lis_module
use precision
use spmdMod,only : masterproc
```

5.4.1 isMonthlyAlarmRinging (Source File: listime_mgr.F90)

INTERFACE:

```
interface isMonthlyAlarmRinging
```

PRIVATE MEMBER FUNCTIONS:

```
module procedure isMonthlyAlarmRinging1
module procedure isMonthlyAlarmRinging2
```

DESCRIPTION:

checks if the monthly alarm is ringing. The private functions have different arguments based on the input options specified.

5.4.2 timemgr_init (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine timemgr_init(lis)
  implicit none
```

ARGUMENTS:

```
  type(lisdec) :: lis
```

DESCRIPTION:

Initialize the LIS time manager.

NOTE - This assumes that the LIS time specific variables pertaining to start and end times have been set before this routine is called.

lis instance of the **lis_module**

The calling sequence is:

date2time (5.4.20)

to convert current date to a floating point format

timemgr_print display the contents of the time manager

5.4.3 timemgr_set (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine timemgr_set(lis, yr, mo, da, hr, mn, ss)
```

USES:

```
use lis_logmod, only : logunit
  implicit none
```

ARGUMENTS:

```
type(lisdec) :: lis
integer :: yr,mo,da,hr,mn,ss
```

DESCRIPTION:

sets the time manager clock based on the input time specification. This method is used to initialize the time manager when the clock is passed down to LIS from a parent component

lis instance of the `lis_module`

yr year

mn month

da day of the month

hr hour of the day

mn minute of the hour

ss second

The calling sequence is:

date2time (5.4.20)
convert date to a floating point format

5.4.4 timemgr_print (Source File: listime_mngr.F90)

INTERFACE:

```
subroutine timemgr_print(lis)
```

5.4.5 advance_timestep (Source File: listime_mngr.F90)

INTERFACE:

```
subroutine advance_timestep(lis)
```

USES:

```
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

```
type(lisdec) :: lis
```

DESCRIPTION:

Increments time manager clock by the model timestep. In case of LIS running multiple nests, each running at different timesteps, the smallest timestep is chosen to advance the clock.

lis instance of the `lis_module`

The calling sequence is:

date2time (5.4.20)
convert date to a floating point format

5.4.6 get_step_size (Source File: listime_mgr.F90)

INTERFACE:

```
function get_step_size(lis)
implicit none
```

ARGUMENTS:

```
type(lisdec) :: lis
integer :: get_step_size
```

DESCRIPTION:

Return the timestep used by the clock in the time manager.

lis instance of the `lis_module`
get_step_size timestep value

5.4.7 get_nstep (Source File: listime_mgr.F90)

INTERFACE:

```
function get_nstep(lis,n)
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
type(lisdec) :: lis
integer :: get_nstep
```

DESCRIPTION:

Return the timestep number for each nest.

lis instance of the `lis_module`
n index of the nest
get_nstep timestep number

5.4.8 get_curr_day (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine get_curr_date(lis, yr, mon, day, tod, offset)
implicit none
```

ARGUMENTS:

```
type(lisdec) :: lis
integer, intent(out) :: yr, mon, day, tod
integer, optional, intent(in) :: offset
```

DESCRIPTION:

Return date components valid at end of current timestep with an optional offset (positive or negative) in seconds.

The arguments are:

lis instance of the `lis_module`

yr year

mon month

day day of the month

tod time of day (seconds past 0Z)

offset offset from current time in seconds positive for future times, negative for previous times.

5.4.9 get_julhr (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine get_julhr(yr, mo, da, hr, mn, ss, julhr)
implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: yr
integer, intent(in)      :: mo
integer, intent(in)      :: da
integer, intent(in)      :: hr
integer, intent(in)      :: mn
integer, intent(in)      :: ss
integer                  :: julhr
```

DESCRIPTION:

Returns the julian hour. In this convention, julian day began at midnight at the beginning of May 24, 1968. Interestingly, this convention was introduced by NASA for the space program.

The arguments are:

yr year

mo month

da day of the month

hr hour of day

mn minute

ss second

julhr julian hour

5.4.10 get_curr_calday (Source File: listime_mgr.F90)

INTERFACE:

```
function get_curr_calday(lis,offset)
    implicit none
```

ARGUMENTS:

```
type(lisdec) :: lis
integer, optional, intent(in) :: offset
real(r8) :: get_curr_calday
```

DESCRIPTION:

Return calendar day at end of current timestep with optional offset. Calendar day 1.0 = 0Z on Jan 1.
The arguments are:

lis instance of the `lis_module`

offset offset from current time in seconds, positive for future times, negative for previous times.

get_curr_calday calendar day value

5.4.11 is_last_step (Source File: listime_mgr.F90)

INTERFACE:

```
function is_last_step(lis)
    implicit none
```

ARGUMENTS:

```
type(lisdec) :: lis
logical :: is_last_step
```

DESCRIPTION:

Function returns true on last timestep.

lis instance of the `lis_module`

is_last_step result of the function

5.4.12 setMonthlyAlarm (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine setMonthlyAlarm(alarmTime)

    implicit none
```

ARGUMENTS:

```
    real*8 :: alarmTime
```

DESCRIPTION:

Initializes the monthly alarm

The arguments are:

alarmTime time of the monthly alarm

5.4.13 isMonthlyAlarmRinging1 (Source File: listime_mgr.F90)

INTERFACE:

```
!Private name: call using isMonthlyAlarmRinging
subroutine isMonthlyAlarmRinging1(lis, alarmTime, interval, &
    midmonth, ringFlag)

    implicit none
```

ARGUMENTS:

```
type(lisdec), intent(in) :: lis
real*8, intent(inout) :: alarmTime
logical, intent(inout):: ringFlag
integer, intent(in) :: interval
logical, intent(in) :: midmonth
```

DESCRIPTION:

checks if the monthly alarm is ringing. The function returns true when the elapsed alarm time is greater than the number of months specified in the alarm's interval. If the midmonth flag is enabled, the elapsed time is counted from the middle of the month to middle of another month, rather than from the beginning of the month to the end of another month.

The arguments are:

lis instance of the **lis_module**

alarmTime the elapsed alarm time

interval the alarm's frequency

midmonth flag to indicate if the elapsed time is to be counted from the beginning or the middle of the month

ringflag flag indicating the status of the call

The calling sequence is:

date2time (5.4.20)

convert date to a floating point format

5.4.14 isMonthlyAlarmRinging2 (Source File: listime_mgr.F90)

INTERFACE:

```
!Private name: call using isMonthlyAlarmRinging
subroutine isMonthlyAlarmRinging2(lis, alarmTime, interval, ringFlag)
```

5.4.15 setHourlyAlarm (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine setHourlyAlarm(alarmTime)
```

```
    implicit none
```

ARGUMENTS:

```
    real*8 :: alarmTime
```

DESCRIPTION:

Initializes the hourly alarm

The arguments are:

alarmTime time of the monthly alarm

5.4.16 isHourlyAlarmRinging (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine isHourlyAlarmRinging(lis, alarmTime, interval, ringFlag)
```

```
    implicit none
```

ARGUMENTS:

```
    type(lisdec), intent(in) :: lis
    real*8, intent(inout) :: alarmTime
    integer, intent(in) :: interval
    logical, intent(inout) :: ringFlag
```

DESCRIPTION:

checks if the hourly alarm is ringing. The function returns true when the elapsed alarm time is greater than the number of hours specified in the alarm's interval.

The arguments are:

lis instance of the **lis_module**

alarmTime the elapsed alarm time

interval the alarm's frequency in hours

ringflag flag indicating the status of the call

The calling sequence is:

tick (5.4.22)

```
    advance the time by the specified increment
```

5.4.17 computeTimeBookEnds (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine computeTimeBookEnds(lis,interval,time1,time2)
    implicit none
```

ARGUMENTS:

```
type(lisdec), intent(in) :: lis
integer, intent(in) :: interval
real*8, intent(inout) :: time1,time2
```

DESCRIPTION:

compute the time end points based on the interval. If the interval is 3 hours, and the current time is 1:30Z, time1 will be set to 1Z and time2 will be set to 4Z.

The arguments are:

lis instance of the `lis_module`

interval time interval in hours

time1 previous time

time2 next time

The calling sequence is:

tick (5.4.22)
advance the time by the specified increment

5.4.18 computeMonthlyWeights (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine computeMonthlyWeights(lis,interval,mo1,mo2,wt1,wt2)
    implicit none
```

ARGUMENTS:

```
type(lisdec) :: lis
integer :: interval
integer :: mo1,mo2
real      :: wt1,wt2
```

DESCRIPTION:

This routine computes the time interpolation weights based on the climatology interval, current time, and the previous and next month. For example, a monthly climatology data to the current date can be interpolated as $\text{value} = \text{value1} * \text{wt1} + \text{value2} * \text{wt2}$ where value1 is the value from the previous month's climatology, and value2 is the value from the next month's climatology and value is the interpolated value.

The arguments are:

lis instance of the `lis_module`

interval interval of the climatology

mo1 previous month

mo2 next month

wt1, wt2 interpolation weights to be used on the climatology data to interpolate to the current day.

The calling sequence is:

date2time (5.4.20)

converts date to a floating point format

5.4.19 computeMonthlyWeights (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine computeMonthlyWeights(lis,interval,mo1,mo2,wt1,wt2)  
    implicit none
```

ARGUMENTS:

```
type(lisdec) :: lis  
integer :: interval  
integer :: mo1,mo2  
real :: wt1,wt2
```

DESCRIPTION:

This routine computes the time interpolation weights based on the climatology interval, current time, and the previous and next month. For example, a monthly climatology data to the current date can be interpolated as $\text{value} = \text{value1} * \text{wt1} + \text{value2} * \text{wt2}$ where value1 is the value from the previous month's climatology, and value2 is the value from the next month's climatology and value is the interpolated value.

The arguments are:

lis instance of the `lis_module`

interval interval of the climatology

mo1 previous month

mo2 next month

wt1, wt2 interpolation weights to be used on the climatology data to interpolate to the current day.

The calling sequence is:

date2time (5.4.20)

converts date to a floating point format

5.4.20 date2time (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine date2time(time,doy,gmt,yr,mo,da,hr,mn,ss)
implicit none
```

ARGUMENTS:

```
integer :: yr,mo,da,hr,mn,ss, doy
real*8  :: time
real     :: gmt
```

DESCRIPTION:

determines the time, time in GMT, and the day of the year based on the value of year, month, day of month, hour of the day, minute and second. This method is the inverse of time2date
The arguments are:

yr year
mo month
da day of the month
hr hour of day
mn minute
ss second
time lis time
gmt time in GMT
doy day of the year

5.4.21 time2date (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine time2date(time,doy,gmt,yr,mo,da,hr,mn)
implicit none
```

ARGUMENTS:

```
integer :: yr,mo,da,hr,mn,ss,doy
real*8  :: time
real     :: gmt
```

DESCRIPTION:

determines the value of the year, month, day of month, hour of the day, minute and second based on the specified time. This method is the inverse of date2time
The arguments are:

yr year
mo month
da day of the month
hr hour of day
mn minute
ss second
time lis time
gmt time in GMT
doy day of the year

5.4.22 tick (Source File: listime_mgr.F90)

INTERFACE:

```
subroutine tick(time,doy,gmt,yr,mo,da,hr,mn,ss,ts)
    implicit none
```

ARGUMENTS:

```
real*8 :: time
integer :: yr,mo,da,hr,mn,ss,ts,doy
real :: gmt
```

DESCRIPTION:

Method to advance or retract the time by the specified time increment
The arguments are:

yr year
mo month
da day of the month
hr hour of day
mn minute
ss second
ts time increment in seconds
time lis time
gmt time in GMT
doy day of the year

5.4.23 julhr_date (Source File: listime_mgr.F90)

REVISION HISTORY:

```
15 oct 1998 initial version.....mr moore/dnxm
10 aug 1999 ported to ibm sp2. added intent attributes to
arguments.....mr gayno/dnxm
29 oct 2005 Sujay Kumar, Adopted in LIS
```

INTERFACE:

```
subroutine julhr_date( julhr, yyyy,mm,dd,hh)
```

USES:

```
use lis_logmod, only : lis_abort  
implicit none
```

ARGUMENTS:

```
integer,      intent(in)  :: julhr
integer          :: yyyy
integer          :: mm
integer          :: dd
integer          :: hh
```

DESCRIPTION:

to convert from a julian hour to a 10 digit date/time group (yyyymmddhh)

5.4.24 Method

: call utility routine tmjul4 to convert from julian hours to year, month, day, hour.
perform several checks to ensure date information passed back from tmjul4 is valid.
if date is good, convert from integer data to a 10 digit date/time group and pass back to calling routine.

The arguments are:

julhr input julian hour

yyyy four digit year

mm month of the year

dd day of the month

hh time of the day in hours

The calling sequence is:

tmjul4 (5.4.25)
convert julian hour to hour, day, month, and year

lis_abort (5.24.2)
abort the code in case of error

5.4.25 tmjul4 (Source File: listime_mgr.F90)

REVISION HISTORY:

```
18 mar 98 initial version.....sra milburn/dnxm
08 jul 99 fixed error which initialized done flag in a data
          data statement. variables must be initialized
          using an assignment statement. ported to ibm sp2.....
          .....mr gayno/dnxm
29 oct 2005 Sujay Kumar, Adopted in LIS
```

INTERFACE:

```
subroutine tmjul4( hour, day, month, year, julhr )
implicit none
```

ARGUMENTS:

```
integer, intent(out)      :: day
integer, intent(out)      :: hour
integer, intent(in)       :: julhr
integer, intent(out)      :: month
integer, intent(out)      :: year
```

DESCRIPTION:

uses the julian hour to determine the 2-digit zulu time, day, month, and 4-digit year.

5.4.26 Method

- determine the current zulu hour
- determine the total number of elapsed days
- count forward to the current day/month/year

The arguments are:

hour the zulu time of the julian hour

day day of the month (1..31)

month month of the year (1..12)

year four digit year

julhr the julian hour being processed

5.5 Fortran: Module Interface spmdMod (Source File: spmdMod.F90)

The primary mode of parallel processing in LIS uses the SPMD style, which refers to "single program, multiple data". In this mode, all processors use the same program, though each has its own data. The SPMD mode is enabled through the use of Message Passing Interface (MPI) in LIS.

This module provides variables and routines to be used in a parallel processing environment. This includes variables to keep track of processor resources and domain decomposition information.

USES:

```
#if (defined SPMD)
    use mpishorthand
#endif
```

PUBLIC MEMBER FUNCTIONS:

```
public :: spmd_init      ! initialize variables
public :: spmd_setup      ! allocate memory
public :: spmd_finalize   ! cleanup allocated structures
```

PUBLIC TYPES:

```
public :: masterproc ! =0 if the current processor is the master
public :: iam         ! current processor's id
public :: npes        ! total number of processors
public :: c_str       ! starting E-W index of each nest
public :: c_end       ! ending E-W index of each nest
public :: r_str       ! starting N-S index of each nest
public :: r_end       ! ending N-S index of each nest
public :: deltas      ! size of unmasked grid space buffers
public :: offsets     ! offsets unmasked grid space buffers
public :: tdeltas     ! size of tile space buffers
public :: toffsets    ! offsets of tile space buffers
public :: gdeltas     ! size of grid space buffers
public :: goffsets    ! offsets of grid space buffers
public :: nchs        ! size of tile spaces
public :: ngrids      ! size of grid spaces
```

5.5.1 spmd_init (Source File: spmdMod.F90)

INTERFACE:

```
interface spmd_init
```

PRIVATE MEMBER FUNCTIONS:

```
module procedure spmd_init_offline
module procedure spmd_init_coupled
```

DESCRIPTION:

This interface initializes the MPI variables, routines for SPMD style of parallel processing. The private routines handle the differences in initializations in an offline and a coupled mode.

5.5.2 spmd_init_offline (Source File: spmdMod.F90)

INTERFACE:

```
subroutine spmd_init_offline()
```

DESCRIPTION:

Initializes MPI and retrieves the number of CPUs and the processors IDs. The MPI initialization is done in an offline mode. In a coupled mode, the initialized environment is passed to LIS from a parent component.

5.5.3 spmd_init_couple (Source File: spmdMod.F90)**INTERFACE:**

```
subroutine spmd_init_couple(nprocs)
```

DESCRIPTION:

Obtains the number of processors and the processors IDs from an already initialized MPI state. The initialization is assumed to be performed in a parent component.

5.5.4 spmd_setup (Source File: spmdMod.F90)**INTERFACE:**

```
subroutine spmd_setup(nnest)
implicit none
```

ARGUMENTS:

```
integer, intent(in):: nnest
```

DESCRIPTION:

Allocates memory for the variables describing domain decomposition.
The arguments are:

nnest Number of nests or domains

5.5.5 spmd_finalize (Source File: spmdMod.F90)**INTERFACE:**

```
subroutine spmd_finalize
```

DESCRIPTION:

This routine issues the invocation to deallocate and cleanup any allocated data structures.

5.6 Fortran: Module Interface output_metaMod (Source File: output_metaMod.F90)

This module is used by the user to define the metadata associated with model output. The module lists a superset of the land surface model variables. The user can choose a subset from this list through the lis configuration file for model output. Currently this list includes the variable definitions from ALMA specification.

<http://www.lmd.jussieu.fr/ALMA/>

REVISION HISTORY:

21 Oct 2005 Sujay Kumar Initial Specification

```
integer :: swnet      ! Net shortwave radiation (surface) (W/m2)
integer :: lwnet      ! Net longwave radiation (surface) (W/m2)
integer :: qle        ! Latent Heat Flux (W/m2)
integer :: qh         ! Sensible Heat Flux (W/m2)
integer :: qg         ! Ground Heat Flux (W/m2)
integer :: qf         ! Energy of fusion (W/m2)
integer :: qv         ! Energy of sublimation (W/m2)
integer :: qtau       ! Momentum flux (N/m2)
integer :: qa         ! Advection flux (W/m2)
integer :: delsurfheat ! Change in surface heat storage (J/m2)
integer :: delcoldcont ! Change in snow water content (J/m2)

integer :: snowf       ! Snowfall rate (kg/m2s)
integer :: rainf       ! Rainfall rate (kg/m2s)
integer :: evap        ! Evapotranspiration (kg/m2s)
integer :: qs          ! Surface Runoff(kg/m2s)
integer :: qrec        ! Recharge from river to the floodplain (kg/m2s)
integer :: qsb         ! Subsurface Runoff (kg/m2s)
integer :: qsm         ! Snowmelt (kg/m2s)
integer :: qst         ! Snow throughfall (kg/m2s)
integer :: delsoilmoist ! DelSoilMoist
integer :: delswe       ! DelSWE
integer :: delsurfstor ! Change in surface water storage (kg/m2)
integer :: delintercept ! Change in interception storage (kg/m2)

integer :: snowt        ! Snow surface temperature (K)
integer :: vegt         ! Vegetation canopy temperature (K)
integer :: baresoilt    ! Temperature of bare soil (K)
integer :: avgsurft     ! Average Surface Temperature (K)
integer :: radt         ! Surface Radiative Temperature (K)
integer :: albedo        ! Surface Albedo (-)
integer :: swe           ! Snow water equivalent (kg/m2)
integer :: sveveg        ! SWE intercepted by vegetation (kg/m2)
integer :: surfstor      ! Surface water storage (kg/m2)

integer :: soilmoist
integer :: soiltemp
integer :: smliqfrac   ! Average layer fraction of liquid
                      ! moisture
integer :: smfrozfrac  ! Average layer fraction of liquid
                      ! moisture

integer :: soilwet      ! Total Soil Wetness (-)
integer :: potevap      ! Potential Evapotranspiration (kg/m2s)
```

```

integer :: ecanop      ! Interception evaporation (kg/m2s)
integer :: tveg        ! Vegetation transpiration (kg/m2s)
integer :: esoil        ! Bare soil evaporation (kg/m2s)
integer :: ewater       ! Open water evaporation (kg/m2s)
integer :: rootmoist    ! Root zone soil moisture (kg/m2)
integer :: canopint     ! Total canopy water storage (kg/m2s)
integer :: evapsnow      ! Snow evaporation (kg/m2s)
integer :: subsnow        ! Snow sublimation (kg/m2s)
integer :: subsurf       ! Sublimation of the snow free area (kg/m2s)
integer :: acond         ! Aerodynamic conductance (m/s)

integer :: etpndx       ! Ponded water evaporation (kg/m2s)
integer :: infxsrt      ! Infiltration excess (kg/m2)
integer :: sfcheadrt     ! Surface overland flow head (kg/m2)

```

This file contains the Config class definition and all Config class methods. This module is adopted from the ESMF source and renamed with LIS names to avoid conflict at a later stage.

5.7 Fortran: Module Interface LIS_ConfigMod - Implements LIS configuration management (Source File: LIS_ConfigMod.F90)

The code in this file implements the LIS_Config class that implements the configuration management system.

5.7.1 Package Overview

LIS Configuration Management is based on NASA DAO's Inpak 90 package, a Fortran 90 collection of routines/functions for accessing *Resource Files* in ASCII format. The package is optimized for minimizing formatted I/O, performing all of its string operations in memory using Fortran intrinsic functions.

Module LIS_ConfigMod is implemented in Fortran 90.

5.7.2 Resource Files

A *Resource File* is a text file consisting of variable length lines (records), each possibly starting with a *label* (or *key*), followed by some data. A simple resource file looks like this:

```

# Lines starting with # are comments which are
# ignored during processing.
my_file_names:      jan87.dat jan88.dat jan89.dat
radius_of_the_earth: 6.37E6 # these are comments too
constants:          3.1415   25
my_favourite_colors: green blue 022 # text & number are OK

```

In this example, my_file_names: and constants: are labels, while jan87.dat, jan88.dat and jan89.dat are data associated with label my_file_names:. Resource files can also contain simple tables of the form,

```

my_table_name:::
1000    3000    263.0
  925    3000    263.0
  850    3000    263.0
  700    3000    269.0
  500    3000    287.0
  400    3000    295.8

```

```

300      3000     295.8
::
```

Resource files are intended for random access (except between ::'s in a table definition). Normally, the order of records should not be important. However, the order of records may be important if the same label appears multiple times.

5.7.3 A Quick Stroll

The first step is to create the LIS Configuration and load the ASCII resource (rc) file into memory¹:

```

config = LIS_ConfigCreate ( rc )
call LIS_ConfigLoadFile (config, filename, layout, rc = rc)
```

Parameter `layout` is optional. If it is passed, multiprocessor performance is optimized. Otherwise, resource file `filename` is read by each processor.

The next step is to select the label (record) of interest, say

```
call LIS_ConfigFindLabel ( config, 'constants:', rc = rc )
```

The 2 constants above can be retrieved with the following code fragment:

```

real      r
integer   i
call LIS_ConfigFindLabel( config, 'constants:', rc = rc)
call LIS_ConfigGetFloat( config, r, rc = rc )           ! results in r = 3.1415
call LIS_ConfigGetInt( config, i, rc = rc )             ! results in i = 25
```

The file names above can be retrieved with the following code fragment:

```

character*20 fn1, fn2, fn3
integer      rc
call LIS_ConfigFindLabel ( config, 'my_file_names:', rc = rc )
call LIS_ConfigGetString ( config, fn1, rc = rc ) ! ==> fn1 = 'jan87.dat'
call LIS_ConfigGetString ( config, fn2, rc = rc ) ! ==> fn1 = 'jan88.dat'
call LIS_ConfigGetString ( config, fn3, rc = rc ) ! ==> fn1 = 'jan89.dat'
```

To access the table above, the user first must use `LIS_ConfigFindLabel()` to locate the beginning of the table, e.g.,

```
call {\tt LIS\_ConfigFindLabel(config, 'my_table_name::', rc = rc)}
```

Subsequently, `callLIS_ConfigNextLine()` can be used to gain access to each row of the table. Here is a code fragment to read the above table (7 rows, 3 columns):

```

real          table(7,3)
character*20  word
integer       rc
call LIS_ConfigFindLabel(config, 'my_table_name::', rc = rc)
do i = 1, 7
    call LIS_ConfigNextLine( config, rc = rc )
```

¹See next section for a complete description of parameters for each routine/function

```

do j = 1, 3
    call LIS_ConfigGetFloat( config, table(i, j), rc = rc )
end do
end do

```

The work with the configuration `config` is finalized by call to `LIS_ConfigDestroy()`:

```

integer rc
call LIS_ConfigDestroy( config, rc )

```

Common Arguments:

character(*) ::	filename	file name
integer ::	rc	error return code (0 is OK)
character(*) ::	label	label (key) to locate record
character(*) ::	word	blank delimited string
character(*) ::	string	a sequence of characters

See the Prologues in the next section for additional details.

5.7.4 Package History

The LIS Configuration Management Package was evolved by Leonid Zaslavsky and Arlindo da Silva from Ipack90 package created by Arlindo da Silva at NASA DAO.

Back in the 70's Eli Isaacson wrote IOPACK in Fortran 66. In June of 1987 Arlindo da Silva wrote Inpak77 using Fortran 77 string functions; Inpak 77 is a vastly simplified IOPACK, but has its own goodies not found in IOPACK. Inpak 90 removes some obsolete functionality in Inpak77, and parses the whole resource file in memory for performance.

REVISION HISTORY:

```

2apr2003 Leonid Zaslavsky Created from m_inpak90.F90
1may2003 Leonid Zaslavsky Corrected version

```

USES:

```

use lis_logmod
implicit none
private

```

PUBLIC MEMBER FUNCTIONS:

```

public :: LIS_ConfigCreate      ! creates configuration
public :: LIS_ConfigDestroy    ! destroys configuration
public :: LIS_ConfigLoadFile   ! loads resource file into memory
public :: LIS_ConfigFindLabel  ! selects a label (key)
public :: LIS_ConfigNextLine   ! selects next line (for tables)
public :: LIS_ConfigGetAttribute ! returns next value
public :: LIS_ConfigGetChar     ! returns only a single character
public :: LIS_ConfigGetLen      ! gets number of words in the line(function)
public :: LIS_ConfigGetDim       ! gets number of lines in the table
                                ! and max number of columns by word
                                ! counting disregarding type (function)

```

PUBLIC TYPES:

```
public :: LIS_Config
```

5.7.5 LIS_ConfigGetAttribute - Get an attribute from a Config

INTERFACE:

```
interface LIS_ConfigGetAttribute
```

PRIVATE MEMBER FUNCTIONS:

```
module procedure LIS_ConfigGetString
module procedure LIS_ConfigGetFloatR4
module procedure LIS_ConfigGetFloatR8
module procedure LIS_ConfigGetFloatsR4
module procedure LIS_ConfigGetFloatsR8
module procedure LIS_ConfigGetIntI4
module procedure LIS_ConfigGetIntI8
module procedure LIS_ConfigGetIntsI4
module procedure LIS_ConfigGetIntsI8
```

DESCRIPTION:

This interface provides an entry point for getting items from an LIS_Config object.

5.7.6 LIS_ConfigCreate - Create a Config object

INTERFACE:

```
type(LIS_Config) function LIS_ConfigCreate( rc )
```

ARGUMENTS:

```
integer,intent(out), optional :: rc
```

DESCRIPTION:

Creates an LIS_Config for use in subsequent calls.

The arguments are:

[rc] Return code; equals 0 if there are no errors.

5.7.7 LIS_ConfigDestroy - Destroy a Config object

INTERFACE:

```
subroutine LIS_ConfigDestroy( config, rc )
```

ARGUMENTS:

```
  type(LIS_Config), intent(inout) :: config  
  integer,intent(out), optional    :: rc
```

DESCRIPTION:

Destroys the **config** object.

The arguments are:

config Already created LIS_Config object.

[rc] Return code; equals 0 if there are no errors.

5.7.8 LIS_ConfigFindLabel

INTERFACE:

```
subroutine LIS_ConfigFindLabel( config, label, rc )
```

ARGUMENTS:

```
  type(LIS_Config), intent(inout) :: config  
  character(len=*), intent(in)   :: label  
  integer, intent(out), optional :: rc
```

DESCRIPTION:

Finds the **label** (key) in the **config** file.

Since the search is done by looking for a word in the whole resource file, it is important to use special conventions to distinguish labels from other words in the resource files. The DAO convention is to finish line labels by : and table labels by ::.

The arguments are:

config Already created LIS_Config object.

label Identifying label.

[rc] Return code; equals 0 if there are no errors. Equals -1 if buffer could not be loaded, -2 if label not found, and -3 if invalid operation with index.

5.7.9 LIS_ConfigGetAttribute - Get a character string

INTERFACE:

```
! Private name; call using LIS_ConfigGetAttribute()  
subroutine LIS_ConfigGetString( config, value, label, default, rc )
```

ARGUMENTS:

```

type(LIS_Config), intent(inout)      :: config
character(len=*), intent(out)        :: value
character(len=*), intent(in), optional :: label
character(len=*), intent(in), optional :: default
integer, intent(out), optional       :: rc

```

DESCRIPTION:

Gets a sequence of characters. It will be terminated by the first white space.
The arguments are:

config Already created LIS_Config object.

value Returned value.

[label] Identifying label.

[default] Default value if **label** is not found in **config** object.

[rc] Return code; equals 0 if there are no errors.

5.7.10 LIS_ConfigGetAttribute - Get a 4-byte real number

INTERFACE:

```

! Private name; call using LIS_ConfigGetAttribute()
subroutine LIS_ConfigGetFloatR4( config, value, label, default, rc )

```

ARGUMENTS:

```

type(LIS_Config), intent(inout)      :: config
real, intent(out)        :: value
character(len=*), intent(in), optional :: label
real, intent(in), optional      :: default
integer, intent(out), optional   :: rc

```

DESCRIPTION:

Gets a 4-byte real **value** from the **config** object.
The arguments are:

config Already created LIS_Config object.

value Returned value.

[label] Identifying label.

[default] Default value if **label** is not found in **config** object.

[rc] Return code; equals 0 if there are no errors.

5.7.11 LIS_ConfigGetAttribute - Get an 8-byte real number

INTERFACE:

```
! Private name; call using LIS_ConfigGetAttribute()
subroutine LIS_ConfigGetFloatR8( config, value, label, default, rc )
```

ARGUMENTS:

```
type(LIS_Config), intent(inout)      :: config
real*8, intent(out)      :: value
character(len=*) , intent(in), optional :: label
real, intent(in), optional      :: default
integer, intent(out), optional    :: rc
```

DESCRIPTION:

Gets an 8-byte real value from the **config** object.

The arguments are:

config Already created LIS_Config object.

value Returned real value.

[label] Identifying label.

[default] Default value if **label** is not found in **config** object.

[rc] Return code; equals 0 if there are no errors.

5.7.12 LIS_ConfigGetAttribute - Get a list of 4-byte real numbers

INTERFACE:

```
! Private name; call using LIS_ConfigGetAttribute()
subroutine LIS_ConfigGetFloatsR4( config, valueList, count, label, &
                                 default, rc )
```

ARGUMENTS:

```
type(LIS_Config), intent(inout)      :: config
real, intent(inout)      :: valueList(:)
integer, intent(in)      :: count
character(len=*) , intent(in), optional :: label
real, intent(in), optional      :: default
integer, intent(out), optional    :: rc
```

DESCRIPTION:

Gets a 4-byte real **valueList** of a given **count** from the **config** object.

The arguments are:

config Already created LIS_Config object.

valueList Returned real values.

count Number of returned values expected.

[label] Identifying label.

[default] Default value if label is not found in configuration object.

[rc] Return code; equals 0 if there are no errors.

5.7.13 LIS_ConfigGetAttribute - Get a list of 8-byte real numbers

INTERFACE:

```
! Private name; call using LIS_ConfigGetAttribute()
subroutine LIS_ConfigGetFloatsR8( config, valueList, count, label, &
                                 default, rc )
```

ARGUMENTS:

```
type(LIS_Config), intent(inout)      :: config
real*8, intent(inout)      :: valueList(:)
integer, intent(in)          :: count
character(len=*), intent(in), optional :: label
real, intent(in), optional   :: default
integer, intent(out), optional :: rc
```

DESCRIPTION:

Gets an 8-byte real **valueList** of a given **count** from the **config** object.
The arguments are:

config Already created **LIS_Config** object.

valueList Returned values.

count Number of returned values expected.

[label] Identifying label.

[default] Default value if label is not found in configuration object.

[rc] Return code; equals 0 if there are no errors.

5.7.14 LIS_ConfigGetAttribute - Get a 4-byte integer number

INTERFACE:

```
! Private name; call using LIS_ConfigGetAttribute()
subroutine LIS_ConfigGetIntI4( config, value, label, default, rc )
```

ARGUMENTS:

```
type(LIS_Config), intent(inout)      :: config
integer, intent(out)      :: value
character(len=*), intent(in), optional :: label
integer, intent(in), optional      :: default
integer, intent(out), optional     :: rc
```

DESCRIPTION:

Gets an integer **value** from the **config** object.

The arguments are:

config Already created LIS_Config object.

value Returned integer value.

[label] Identifying label.

[default] Default value if label is not found in configuration object.

[rc] Return code; equals 0 if there are no errors.

5.7.15 LIS_ConfigGetAttribute - Get an 8-byte integer number

INTERFACE:

```
! Private name; call using LIS_ConfigGetAttribute()
subroutine LIS_ConfigGetIntI8( config, value, label, default, rc )
```

ARGUMENTS:

```
type(LIS_Config), intent(inout)      :: config
integer*8, intent(out)      :: value
character(len=*), intent(in), optional :: label
integer, intent(in), optional      :: default
integer, intent(out), optional     :: rc
```

DESCRIPTION:

Gets an 8-byte integer **value** from the **config** object.

The arguments are:

config Already created LIS_Config object.

value Returned integer value.

[label] Identifying label.

[default] Default value if label is not found in configuration object.

[rc] Return code; equals 0 if there are no errors.

5.7.16 LIS_ConfigGetAttribute - Get a list of 4-byte integers

INTERFACE:

```
! Private name; call using LIS_ConfigGetAttribute()
subroutine LIS_ConfigGetIntsI4( config, valueList, count, label, &
                               default, rc )
```

ARGUMENTS:

```
type(LIS_Config), intent(inout)      :: config
integer, intent(inout)    :: valueList(:)
integer, intent(in)          :: count
character(len=*) , intent(in), optional :: label
integer, intent(in), optional      :: default
integer, intent(out), optional     :: rc
```

DESCRIPTION:

Gets a 4-byte integer **valueList** of given **count** from the **config** object.
The arguments are:

config Already created LIS_Config object.

valueList Returned values.

count Number of returned values expected.

[label] Identifying label.

[default] Default value if label is not found in configuration object.

[rc] Return code; equals 0 if there are no errors.

5.7.17 LIS_ConfigGetAttribute - Get a list of 8-byte integers

INTERFACE:

```
! Private name; call using LIS_ConfigGetAttribute()
subroutine LIS_ConfigGetIntsI8( config, valueList, count, label, &
                               default, rc )
```

ARGUMENTS:

```
type(LIS_Config), intent(inout)      :: config
integer*8, intent(inout)    :: valueList(:)
integer, intent(in)          :: count
character(len=*) , intent(in), optional :: label
integer, intent(in), optional      :: default
integer, intent(out), optional     :: rc
```

DESCRIPTION:

Gets an 8-byte integer **valueList** of given **count** from the **config** object.
The arguments are:

config Already created LIS_Config object.

valueList Returned values.

count Number of returned values expected.

[label] Identifying label.

[default] Default value if label is not found in configuration object.

[rc] Return code; equals 0 if there are no errors.

5.7.18 LIS_ConfigGetChar - Get a character

INTERFACE:

```
subroutine LIS_ConfigGetChar( config, value, label, default, rc )
```

ARGUMENTS:

```
type(LIS_Config), intent(inout)      :: config
character, intent(out)                :: value
character(len=*) , intent(in), optional :: label
character, intent(in), optional       :: default
integer, intent(out), optional        :: rc
```

DESCRIPTION:

Gets a character **value** from the **config** object.

The arguments are:

config Already created LIS_Config object.

value Returned value.

[label] Identifying label.

[default] Default value if label is not found in configuration object.

[rc] Return code; equals 0 if there are no errors.

5.7.19 LIS_ConfigGetDim - Get table sizes

INTERFACE:

```
subroutine LIS_ConfigGetDim( config, label, lineCount, columnCount, rc )

implicit none

type(LIS_Config), intent(inout)      :: config      ! LIS Configuration
integer, intent(out)                  :: lineCount
integer, intent(out)                  :: columnCount
```

```

character(len=*), intent(in), optional :: label ! label (if present)
                                         ! otherwise, current
                                         ! line
integer, intent(out), optional           :: rc      ! Error code

```

DESCRIPTION:

Returns the number of lines in the table in `lineCount` and the maximum number of words in a table line in `columnCount`.

The arguments are:

config Already created `LIS_Config` object.

lineCount Returned number of lines in the table.

columnCount Returned maximum number of words in a table line.

[label] Identifying label.

[rc] Return code; equals 0 if there are no errors.

5.7.20 LIS_ConfigGetLen - Get the length of the line in words

INTERFACE:

```
integer function LIS_ConfigGetLen( config, label, rc )
```

ARGUMENTS:

```

type(LIS_Config), intent(inout)      :: config
character(len=*), intent(in), optional :: label
integer, intent(out), optional       :: rc

```

DESCRIPTION:

Gets the length of the line in words by counting words disregarding types. Returns the word count as an integer.

The arguments are:

config Already created `LIS_Config` object.

[label] Identifying label. If not specified, use the current line.

[rc] Return code; equals 0 if there are no errors.

5.7.21 LIS_ConfigNextLine - Find next line

INTERFACE:

```
subroutine LIS_ConfigNextLine( config, tableEnd, rc)
```

ARGUMENTS:

```
type(LIS_Config), intent(inout) :: config
logical, intent(out), optional :: tableEnd
integer, intent(out), optional:: rc
```

DESCRIPTION:

Selects the next line (for tables).

The arguments are:

config Already created LIS_Config object.

[tableEnd] If specified as TRUE, end of table mark (::) is checked.

[rc] Return code; equals 0 if there are no errors.

5.7.22 LIS_ConfigLoadFile - Load resource file into memory

INTERFACE:

```
subroutine LIS_ConfigLoadFile( config, filename, unique, rc )
```

ARGUMENTS:

```
type(LIS_Config), intent(inout) :: config
character(len=*), intent(in)      :: filename
logical, intent(in), optional    :: unique
integer, intent(out), optional   :: rc
```

DESCRIPTION:

Resource file with **filename** is loaded into memory.

The arguments are:

config Already created LIS_Config object.

filename Configuration file name.

[unique] If specified as true, uniqueness of labels are checked and error code set if duplicates found.

[rc] Return code; equals 0 if there are no errors.

5.7.23 LIS_ConfigLoadFile_1proc - Load resource file into memory

INTERFACE:

```
subroutine LIS_ConfigLoadFile_1proc_( config, filename, unique, rc )
```

```
implicit none
```

```
type(LIS_Config), intent(inout) :: config      ! LIS Configuration
character(len=*), intent(in)   :: filename       ! file name
logical, intent(in), optional :: unique        ! if unique is present,
```

```

        ! uniqueness of labels
        ! is checked and error
        ! code is set
        ! Error code
        !   0 no error
        ! -98 coult not get unit
        !   number (strange!)
        ! -98 talk to a wizzard
        ! -99 out of memory: increase
        !   NBUF_MAX
        !   other iostat from open
        !   statement.

```

DESCRIPTION:

Resource file filename is loaded is loaded into memory

5.8 Fortran: Module Interface domain_module.F90 (Source File: domain_module.F90)

The code in this file provides interfaces to manages different running domain implementations

5.8.1 Overview

The definition of a domain in LIS is includes two dimensions. A LIS domain is defined by the projection of the running grid and the input data ordering rules. For example, a domain using latlon projection with a SW to NE data ordering is considered different from a domain using latlon projection with NW to SE data ordering. Each input data used with a domain definition is expected to be transformed to the specified projection and data ordering rules through the interfaces defined in the `param.module`.

REVISION HISTORY:

17 Feb 2004 Sujay Kumar Initial Specification

`implicit none`

PUBLIC MEMBER FUNCTIONS:

`public :: LIS_domain_init !initialize specified domains`
`public :: LIS_domain_finalize !cleanup allocated structures`

5.8.2 LIS_domain_init (Source File: domain_module.F90)

INTERFACE:

`subroutine LIS_domain_init`

USES:

```

use lisdrv_module, only : lis
use spmdMod
use domain_pluginMod, only : domain_plugin

```

DESCRIPTION:

This routine invokes the registry that defines the domain implementations, first. This is followed by calling the routines to read the runtime specific parameters for each domain instance. A call to create the domain is issued next, which is expected to generate the grid, tile, and ensemble spaces for each domain. This routine is also expected to perform any domain decomposition needed for parallel processing. Finally, the domain decomposition information is disseminated to individual processors.

The calling sequence is:

domain_plugin (6.2.1)

sets up function table registries for implemented domains

readinput (5.29.4)

invokes the generic method in the registry to read domain specific input options from the configuration file

makedomain (5.29.2)

invokes the generic method in the registry to create the grid and tile spaces.

5.8.3 LIS_domain_finalize (Source File: domain_module.F90)

INTERFACE:

```
subroutine LIS_domain_finalize
```

DESCRIPTION:

This routine issues the invocation to deallocate and cleanup any allocated data structures in the specific instance of the domain implementation.

5.9 Fortran: Module Interface lsm_module (Source File: lsm_module.F90)

The code in this file provides interfaces to manage the operation of different land surface models

5.9.1 Overview

This module defines the interface plugins for the incorporation of different land surface models. These interfaces provide entry points for introducing a new land surface scheme in LIS. The following specific implementations for each LSM are expected to specify methods to initialize and set the LSM specific variables and parameters, provide methods for model simulation, model output, and restart operations. A number of other optional interfaces need to be specified depending on the mode of operation of the LSM. For example, if the LSM is used for a coupled simulation with an atmospheric component, the LIS_lsm_setexport interface needs to be implemented. Similar implementations need to be specified for the use of the LSM in data assimilation applications.

REVISION HISTORY:

14 Nov 2002 Sujay Kumar Initial Specification

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: LIS_lsm_init      ! initialize lsm variables, memory
public :: LIS_setuplsm       ! set land surface parameters
public :: LIS_readrestart   ! read the restart file
public :: LIS_force2tile    ! transfer forcing to model tiles
public :: LIS_lsm_main       ! execute the land model
public :: LIS_lsm_output     ! write model output
public :: setLSMDynParams   ! set the time dependent parameters
public :: LIS_writerestart  ! write the restart file
public :: LIS_lsm_setexport  ! set the variables to be exported to another
                            ! component
public :: LIS_lsm_finalize   ! cleanup allocated structures
```

5.9.2 LIS_force2tile (Source File: lsm_module.F90)

INTERFACE:

```
interface LIS_force2tile
```

PRIVATE MEMBER FUNCTIONS:

```
module procedure LIS_force2tile_offline
module procedure LIS_force2tile_coupled
```

DESCRIPTION:

This interface provides routines for transferring the meteorological forcing to the model tiles. In an offline simulation, the model forcing is read from an previous analysis (typically from a file), and is set in the LIS datastructures. In a coupled mode, LIS obtains the forcing from the atmospheric component and is transferred to the model tiles.

5.9.3 LIS_lsm_init (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_lsm_init()
```

USES:

```
use lisdrv_module, only: lis
use lsm_pluginMod
```

DESCRIPTION:

Interface for initializing the land model. The intialization includes the allocation of memory for LSM specific variables and datastructures and specification of any runtime specific options.

The calling sequence is:

lsm_plugin (6.5.1)

sets up function table registries for implemented land surface models

lsmini (5.34.2)

invokes the generic method in the registry to initialize the land surface model

5.9.4 LIS_SetupLSM (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_SetupLSM
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

The setup interfaces are used for the LSM specification of LSM bparameters. If a parameter falls outside the LIS-specified generic parameter list, the LSM is expected to provide routines for the handling of any input data, specific to those parameters.

The calling sequence is:

lsmsetup (5.34.8)

invokes the generic method in the registry to set up the land surface model

5.9.5 LIS_lsm_main (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_lsm_main(n)
```

USES:

```
use lisdrv_module, only: lis
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This interface provides the entry point to the LSM routines that invokes the land surface model physics. The arguments are:

n index of the nest or domain

The calling sequence is:

readobservations (5.28.4)

invokes the generic method in the registry to read the observations being assimilated

forecast (5.28.8)

invokes the generic method in the registry to perform a forecast step

assimilate (5.28.10)

invokes the generic method in the registry to perform an assimilation step

lsmrun (5.34.4)

invokes the generic method in the registry to run the land surface model

5.9.6 LIS_readrestart (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_readrestart
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

This interface provides the entry point to read LSM specific model restart files.
The calling sequence is:

lsmrestart (5.34.10)

invokes the generic method in the registry to read restart files for the land surface model

5.9.7 LIS_lsm_output (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_lsm_output(n)
```

USES:

```
use lisdrv_module, only: lis
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This interface provides the entry point to write LSM specific model output files.
The arguments are:

n index of the nest or domain

The calling sequence is:

lsmoutput (5.34.12)

invokes the generic method in the registry to write the land surface model output

5.9.8 setLSMDynparams (Source File: lsm_module.F90)

INTERFACE:

```
subroutine setLSMDynparams(n)
```

USES:

```
use lisdrv_module, only : lis
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This interface provides an entry point to update any time dependent land surface model parameters in an LSM.

The arguments are:

n index of the nest or domain

The calling sequence is:

lsmdynsetup (5.34.30)

invokes the generic method in the registry to set up time dependent parameters for the land surface model

5.9.9 LIS_force2tile_offline (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_force2tile_offline(n)
```

USES:

```
use lisdrv_module, only: lis, lisdom
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This interface is used to transfer the forcing variables to the actual model tile space. Any forcing perturbations that need to be applied to the input forcing is applied at this stage as well.

The arguments are:

n index of the nest or domain

The calling sequence is:

perturb (5.35.4)

invokes the generic method in the registry to perturb the forcing

getpertforcing (5.35.6)

invokes the generic method in the registry to obtain the perturbed forcing

lsmf2t (5.34.14)

invokes the generic method in the registry to transfer the forcing to the land surface model tiles

5.9.10 LIS_force2tile_coupled (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_force2tile_coupled(n,forcing)
```

USES:

```
use lisdrv_module, only: lis, lisdom
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout):: forcing(lis%nch(n),lis%nf)
```

DESCRIPTION:

Transfers grid forcing to model tiles in a coupled run. The forcing is obtained from the atmospheric component. The arguments are:

n index of the nest or domain

forcing the array of meteorological forcing variables

The calling sequence is:

lsmf2t (5.34.14)

invokes the generic method in the registry to transfer the forcing to the land surface model tiles

5.9.11 LIS_writterestart (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_writterestart(n)
```

USES:

```
use lisdrv_module, only : lis
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This interface provides the entry point to read the LSM specific model restart files. The arguments are:

n index of the nest or domain

The calling sequence is:

lsmwrst (5.34.16)

invokes the generic method in the registry to write restart files for the land surface model

5.9.12 LIS_lsm_setexport (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_lsm_setexport(n)
```

USES:

```
use lisdrv_module, only : lis
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This interface provides the entry point for specifying an export state (a list of model specific variables) from a land surface model. The routine is used in a coupled simulation to provide feedback to a different model component such as an atmospheric model.

The arguments are:

n index of the nest or domain

The calling sequence is:

lsmsetexport (5.34.28)

invokes the generic method in the registry to set the export state from the land surface model

5.9.13 LIS_lsm_finalize (Source File: lsm_module.F90)

INTERFACE:

```
subroutine LIS_lsm_finalize()
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

This routine issues the invocation to deallocate and cleanup any allocated data structures in the specific instance of a land surface model

The calling sequence is:

lsmfinalize (5.34.6)

invokes the generic method in the registry to cleanup the LSM related datastructures

5.10 Fortran: Module Interface baseforcing_module (Source File: baseforcing_module.F90)

The code in this file provides interfaces to manage different meteorological forcing analyses

5.10.1 Overview

This module defines interface plugins for the incorporation of various meteorological forcing analyses. The analyses implemented by extending these interfaces should contain all (and possibly more) of the basic meteorological forcing variables. Analyses containing one or more of the basic variables, but not the entire set, should be implemented as a supplemental forcing. The following is a list of the basic meterological forcing variables.

- T 2m: Temperature interpolated to 2m (K)
- Q 2m: Instantaneous specific humidity interpolated to 2m (kg/kg)
- SWdown: Downward shortwave flux at the ground (W/m²)
- LWdown: Downward longwave flux at the ground (W/m²)
- U 10m: Instantaneous zonal wind interpolated to 10m (m/s)
- V 10m: Instantaneous meridional wind interpolated to 10m (m/s)
- Psurf: Instantaneous surface pressure (Pa)
- Precip: Total precipitation (mm/s)

REVISION HISTORY:

14 Nov 2002 Sujay Kumar Initial Specification

implicit none

PUBLIC MEMBER FUNCTIONS:

```
public :: LIS_baseforcing_init      !initialize base forcing setup
public :: LIS_get_base_forcing     !retrieve data and interpolate
                                    !spatially and temporally
public :: LIS_baseforcing_finalize !cleanup allocated structures
```

PUBLIC TYPES:

```
public :: lisforc
```

5.10.2 LIS_baseforcing_init (Source File: baseforcing_module.F90)

INTERFACE:

```
subroutine LIS_baseforcing_init
```

USES:

```
use lisdrv_module, only: lis
use baseforcing_pluginMod
```

DESCRIPTION:

This routine sets up the structures to include meteorological forcing analyses. The registry that defines the implemented forcing schemes are invoked followed by the routines to initialize the specific instance of the forcing scheme.

The methods invoked are:

baseforcing_plugin (6.3.1)

sets up function table registries for implemented base forcing analyses

allocate_forcing_mem (5.10.5)

allocate memory for required structures

defineNative (5.30.6)

invokes the generic method in the registry to define the native domain of the base forcing scheme

5.10.3 LIS_get_base_forcing (Source File: **baseforcing_module.F90**)

INTERFACE:

```
subroutine LIS_get_base_forcing(n)
```

USES:

```
use lisdrv_module, only: lis
use spmdMod, only : iam
```

ARGUMENTS:

```
integer,intent(in) :: n
```

DESCRIPTION:

This routine issues the calls to retrieve the forcing variables from the specific forcing scheme. The retrieval call is expected to read the data, perform any spatial interpolation and other transformations needed to grid the data to the running domain and resolution. This invocation is followed by the call to temporally interpolate the data to the current model timestep. The temporal interpolation is performed based on the two consecutive forcing analyses. A zenith angle based interpolation is performed for the temporal disaggregation of downward shortwave radiation. Finally any topographic corrections and downscaling to the variables are applied.

The arguments are:

n index of the domain or nest.

The methods invoked are:

retrieveforcing (5.30.2)

invokes the generic method in the registry to retrieve the base forcing data

timeinterp (5.30.4)

invokes the generic method in the registry to perform temporal interpolation

lapseRateCorrection (5.22.3)

method to apply topographical corrections

microMetCorrection (5.24.6)

method to apply topographical corrections

5.10.4 LIS_baseforcing_finalize (Source File: baseforcing_module.F90)

INTERFACE:

```
subroutine LIS_baseforcing_finalize
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

This routine issues the invocation to deallocate and cleanup any allocated data structures in the specific instance of the forcing scheme.

The methods invoked are:

forcingfinalize (5.30.8)

invokes the generic method in the registry to cleanup the allocated structures for the base forcing scheme.

5.10.5 allocate_forcing_mem (Source File: baseforcing_module.F90)

INTERFACE:

```
subroutine allocate_forcing_mem()
```

USES:

```
use lisdrv_module, only: lis
```

DESCRIPTION:

This subroutine allocates the memory for the baseforcing datastructures based on the number of domains used and the temporal interpolation scheme used.

5.11 Fortran: Module Interface suppforcing_module (Source File: suppforcing_module.F90)

The code in this file provides interfaces to manage supplemental forcing analyses

5.11.1 Overview

The module provides interfaces for incorporating supplemental forcing analyses into LIS. Supplemental forcing is defined as any analysis that includes a subset of the meteorological forcing variables defined in `baseforcing_module`. Examples include the incorporating precipitation analyses from various sources such as CMAP, CMORPH, etc. If a supplemental forcing is provided, the corresponding variables set by the baseforcing is overwritten.

REVISION HISTORY:

14 Nov 2002 Sujay Kumar Initial Specification

USES:

```
use spmdMod
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: LIS_suppforcing_init ! initializes supplemental forcing
public :: LIS_get_supp_forcing ! retrieve supplemental forcing data
                           ! and interpolate spatially and temporally
public :: LIS_suppforcing_finalize !cleanup allocated structures
```

5.11.2 LIS_suppforcing_init (Source File: suppforcing_module.F90)

INTERFACE:

```
subroutine LIS_suppforcing_init
```

USES:

```
use lisdrv_module, only : lis
use suppforcing_pluginMod, only : suppforcing_plugin
```

DESCRIPTION:

This routine sets up the structures to include supplemental forcing analyses. The registry that defines the implemented schemes are invoked, followed by the routines to initialize the specific instance of the forcing scheme.

The methods invoked are:

suppforcing_plugin (6.4.1)

sets up function table registries for implemented supplemental forcing analyses

defineNativeSupp (5.38.2)

invokes the generic method in the registry to define the native domain of the supplemental forcing scheme

5.11.3 LIS_get_supp_forcing (Source File: suppforcing_module.F90)

INTERFACE:

```
subroutine LIS_get_supp_forcing(n)
```

USES:

```
use lisdrv_module, only: lis
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine issues the call to retrieve data from the specified supplemental forcing analysis. The retrieval call is expected to read the data, perform any spatial interpolation and other transformations needed to grid the data to the running domain and resolution. This invocation is followed by the call to temporally interpolate the data to the current model timestep. The temporal interpolation details are similar to the description in `baseforcing module`.

The arguments are:

`n` index of the domain or nest.

The methods invoked are:

getsuppforc (5.38.4)

invokes the generic method in the registry to retrieve the supplemental forcing data

timeinterpsupp (5.38.6)

invokes the generic method in the registry to perform temporal interpolation

5.11.4 LIS_suppforcing_finalize (Source File: suppforcing_module.F90)

INTERFACE:

```
subroutine LIS_suppforcing_finalize
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

This routine issues the invocation to deallocate and cleanup any allocated data structures in the specific instance of the supplemental forcing scheme.

The methods invoked are:

suppforcingfinalize (5.38.8)

invokes the generic method in the registry to cleanup the allocated structures for the supplemental forcing scheme.

5.12 Fortran: Module Interface dataassim_module (Source File: dataassim_module.F90)

The code in this file provides abstractions to manage different data assimilation implementations.

5.12.1 Overview

This module defines some of the interface plugins for the incorporation of different data assimilation implementations. There are three main levels of abstractions identified in a data assimilation implementation:

- Data Assimilation Algorithm: various methods such as direct insertion, extended kalman filter, ensemble kalman filter, etc.
- Observations/Variable being assimilated: this abstraction identifies the source of variables, and the translation of observations to model state variables.
- Land surface model: the model used in the assimilation implementation.

- Perturbation techniques: methods to perturb the forcing and model prognostic variables

A complete implementation would require the interaction between the specific instances of the components listed above. This module includes methods to invoke some of the above.

REVISION HISTORY:

27 Feb 2005 Sujay Kumar; Initial Specification

`implicit none`

PUBLIC MEMBER FUNCTIONS:

```
public :: LIS_dataassim_init      !initialize data assimilation routines
public :: LIS_dataassim_finalize !cleanup allocated structures
```

5.12.2 LIS_dataassim_init (Source File: dataassim_module.F90)

INTERFACE:

`subroutine LIS_dataassim_init`

USES:

```
use lisdrv_module, only : lis
use dataassim_pluginMod
use dataobs_pluginMod
use perturb_pluginMod
```

DESCRIPTION:

The routine performs the initializations required for data assimilation. The registries that define the instances of data assimilation algorithms, observations, and perturbation methods are invoked, followed by the creation of required data structures.

The calling sequence is:

dataassim_plugin (6.7.1)

sets up function table registries for implemented data assimilation algorithms

dataobs_plugin (6.8.1)

sets up function table registries for implemented observation sources to be assimilated

dataassimsetup (5.28.6)

invokes the generic method in the registry to set up the data assimilation algorithm- structures

dataobssetup (5.28.2)

invokes the generic method in the registry to set up the structures to read the observation data, for assimilation

perturb_plugin (6.9.1)

sets up the function table registries for implemented perturbation algorithms

perturbinit (5.35.2)

invokes the generic method in the registry to initialize the perturbation algorithm

5.12.3 LIS_dataassim_finalize (Source File: dataassim_module.F90)

INTERFACE:

```
subroutine LIS_dataassim_finalize
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

This routine issues the invocation to deallocate and cleanup any allocated data structures in the specific instance of the data assimilation implementations.

The calling sequence is:

dafinalize (5.28.12)

invokes the generic method in the registry to cleanup the allocated structures for the data assimilation algorithm

5.13 Fortran: Module Interface param_module (Source File: param_module.F90)

The code in this file provides interfaces to manage different sources of parameter datasets.

5.13.1 Overview

This module contains interface plugins for the incorporating I/O associated with various sources of land surface parameter maps. These interfaces act as the entry points for handing issues such as format, ordering, and projection associated with a dataset. The parameter reading routines are expected to perform all these transformations and convert the data to same grid, domain, and data ordering as the ones used in the running domain of LIS. The **param_module** provides interfaces to incorporate the following sources of parameter maps.

soils sand, silt, clay fractions, soil color, soil texture

albedo climatology and max albedo over deep snow

greenness climatology

Leaf Area Index climatology

Stem Area Index climatology

Topography static elevation, slope, aspect, curvature data

Bottom Temperature static

REVISION HISTORY:

26 Oct 2005 Sujay Kumar Initial Specification

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
-----  
public :: LIS_param_init !initializes structures, read static data  
public :: LIS_setDynParams !read time dependent data  
public :: LIS_param_finalize !cleanup allocated structures
```

5.13.2 LIS_param_init (Source File: param_module.F90)

INTERFACE:

```
subroutine LIS_param_init()
```

USES:

```
use lisdrv_module, only: lis  
use gfrac_module, only : greenness_setup  
use albedo_module, only : albedo_setup  
use soils_module, only : soils_setup,read_soils  
use lai_module, only : lai_setup,read_lai  
use sai_module, only : sai_setup,read_sai  
use snow_module, only : snow_setup
```

DESCRIPTION:

This interface provides the entry point for the initialization of data structures required for reading in parameter datasets. It also invokes routines to read static datasets such as soils.

The calling sequence is:

soils_setup (5.19.2)

call to set up the soil parameters options

read_soils (5.19.3)

call to read the chosen soil parameters

albedo_setup (5.14.2)

call to setup albedo parameter options

greenness_setup (5.15.2)

call to setup greenness parameter options

lai_setup (5.16.2)

call to setup LAI parameter options

sai_setup (5.17.2)

call to setup SAI parameter options

snow_setup (5.18.1)

call to setup snow parameter options

5.13.3 LIS_setDynparams (Source File: param_module.F90)

INTERFACE:

```
subroutine LIS_setDynparams(n)
```

USES:

```
use lisdrv_module, only : lis
use albedo_module, only : read_albedo
use gfrac_module, only : read_greenness
use lai_module, only : read_lai
use sai_module, only : read_sai
use snow_module, only : read_snow
use lsm_module, only : setLSMDynparams
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This interface provides the entry point for routines to update any time dependent land surface parameters. Current implementation includes datasets such as albedo, greenness, LAI, and SAI.

The arguments are:

n index of the domain or nest.

The calling sequence is:

read_greenness (5.15.3)

call to read the greenness data source

read_albedo (5.14.4)

call to read the albedo data source

read_lai (5.16.3)

call to read the LAI data source

read_sai (5.17.3)

call to read the SAI data source

read_snow (5.18.2)

call to read the snow data source

5.13.4 LIS_param_finalize (Source File: param_module.F90)

INTERFACE:

```
subroutine LIS_param_finalize()
```

USES:

```
use lisdrv_module, only : lis
use albedo_module, only : albedo_finalize
use gfrac_module, only : greenness_finalize
use lai_module, only : lai_finalize
use sai_module, only : sai_finalize
use snow_module, only : snow_finalize
```

DESCRIPTION:

This routine issues the invocation to deallocate and cleanup any allocated data structures used in the parameter dataset implementations.

The calling sequence is:

```
albedo_finalize (5.14.5)
    call to cleanup albedo related structures

greenness_finalize (5.15.4)
    call to cleanup greenness related structures

lai_finalize (5.16.4)
    call to LAI related structures

sai_finalize (5.17.4)
    call to SAI related structures
```

5.14 Fortran: Module Interface albedo_module (Source File: albedo_module.F90)

The code in this file implements routines to read various sources of albedo.

5.14.1 Overview

This routines in this module reads two sources of albedo:

- Albedo climatology
- Static, maximum albedo expected over deep snow

This module provides routines to read the albedo climatology data and allows the users to specify the frequency of climatology (in months). The climatological data is temporally interpolated between months to the current simulation date.

REVISION HISTORY:

```
8 Aug 2005: Sujay Kumar; Initial implementation
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: albedo_setup      !allocates memory for required variables
public :: read_albedo       !reads climatological albedo
public :: albedo_finalize   !cleanup allocated structures
```

PUBLIC TYPES:

```
public :: lis_alb           !data structure containing albedo data
```

5.14.2 albedo_setup (Source File: albedo_module.F90)

INTERFACE:

```
subroutine albedo_setup
```

DESCRIPTION:

Allocates memory for data structures for reading in albedo datasets
The routines invoked are:

read_mxsnalb (5.14.3)

method to read the max snow albedo

setMonthlyAlarm (5.4.12)

sets the monthly alarm to read the albedo climatology

USES:

```
use lisdrv_module, only : lis
use listime_mgr, only : setMonthlyAlarm
```

5.14.3 read_mxsnalb (Source File: albedo_module.F90)

INTERFACE:

```
subroutine read_mxsnalb(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Reads the static albedo upper bound over deep snow for each domain.
The arguments are:

n index of the domain or nest.

The routines invoked are:

readmxsnoalb (5.27.2)

Calls the generic method in the registry to read the max snow albedo

5.14.4 read_albedo (Source File: albedo_module.F90)

INTERFACE:

```
subroutine read_albedo(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use listime_mgr, only : isMonthlyAlarmRinging, computeMonthlyWeights
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Reads the snow free albedo climatology data and temporally interpolates it to the current day
The arguments are:

n index of the domain or nest.

The routines invoked are:

isMonthlyAlarmRinging (5.4.1)

checks to see if it is time to read the climatology data

computeMonthlyWeights (5.4.19)

computes the interpolation weights

readalbedo (5.27.4)

invokes the generic method in the registry to read the albedo climatology data

5.14.5 albedo_finalize (Source File: albedo_module.F90)

INTERFACE:

```
subroutine albedo_finalize
```

USES:

```
use lisdrv_module, only : lis
implicit none
```

DESCRIPTION:

Deallocates objects created in this module

5.15 Fortran: Module Interface gfrac_module (Source File: gfrac_module.F90)

The code in this file implements routines to read greenness fraction data.

5.15.1 Overview

This routines in this module provides routines to read the greenness fraction climatology data and allows the users to specify the frequency of climatology (in months). The climatological data is temporally interpolated between months to the current simulation date.

REVISION HISTORY:

```
08 Aug 2005: Sujay Kumar; Initial implementation  
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: greenness_setup      !allocates memory for required structures  
public :: read_greenness      !reads the greenness data  
public :: greenness_finalize  !cleanup allocated structures
```

PUBLIC TYPES:

```
public :: lis_gfrac !data structure containing greenness fraction data.
```

5.15.2 greenness_setup (Source File: gfrac_module.F90)

INTERFACE:

```
subroutine greenness_setup
```

USES:

```
use lisdrv_module, only : lis  
use listime_mgr, only : setMonthlyAlarm
```

DESCRIPTION:

Allocates memory for data structures for reading the greenness fraction datasets
The routines invoked are:

setMonthlyAlarm (5.4.12)
sets the monthly alarm to read the greenness climatology

5.15.3 read_greenness (Source File: gfrac_module.F90)

INTERFACE:

```
subroutine read_greenness(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use listime_mgr, only : isMonthlyAlarmRinging, computeMonthlyWeights
use spmdMod, only : masterproc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Reads the greenness fraction climatology and temporally interpolates it to the current day.
The arguments are:

n index of the domain or nest.

The routines invoked are:

isMonthlyAlarmRinging (5.4.1)

checks to see if it is time to read the climatology data

computeMonthlyWeights (5.4.19)

computes the interpolation weights

readgfrac (5.31.2)

invokes the generic method in the registry to read the greenness climatology data

5.15.4 greenness_finalize (Source File: gfrac_module.F90)

INTERFACE:

```
subroutine greenness_finalize
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

Deallocates objects created in this module

5.16 Fortran: Module Interface lai_module (Source File: lai_module.F90)

The code in this file provides interfaces to manage different sources of Leaf Area Index (LAI) data.

5.16.1 Overview

This routines in this module provides routines to read the LAI climatology data. The climatological data is temporally interpolated between months to the current simulation date.

Note that currently the module assumes the frequency of LAI climatology to be monthly.

REVISION HISTORY:

08 Aug 2005: Sujay Kumar; Initial implementation

USES:

```
use precision
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: lai_setup ! allocates memory for required structures
public :: read_lai   ! reads the LAI data
public :: lai_finalize ! cleanup allocated structures
```

PUBLIC TYPES:

```
public :: lis_lai ! data structure containing LAI data.
```

5.16.2 lai_setup (Source File: lai_module.F90)

INTERFACE:

```
subroutine lai_setup
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

Allocates memory and other structures for reading LAI datasets

5.16.3 read_lai (Source File: lai_module.F90)

INTERFACE:

```
subroutine read_lai(n)
```

USES:

```
use lisdrv_module, only : lis
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Reads the LAI climatology and temporally interpolates it to the current day.
The arguments are:

n index of the domain or nest.

The routines invoked are:

readlai (5.32.2)

invokes the generic method in the registry to read the LAI climatology data

5.16.4 lai_finalize (Source File: lai_module.F90)

INTERFACE:

```
subroutine lai_finalize
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

Deallocates objects created in this module

5.17 Fortran: Module Interface sai_module (Source File: sai_module.F90)

The code in this file provides interfaces to manage different sources of Stem Area Index (SAI) data.

5.17.1 Overview

This module provides routines to ingest the SAI climatology data and allows users to specify the frequency of climatology (in months). The climatological data is then temporally interpolated between months to the current simulation date.

REVISION HISTORY:

08 Aug 2005: Sujay Kumar; Initial implementation

USES:

```
use precision
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
-----  
public :: sai_setup      ! allocates memory for required structures  
public :: read_sai        ! reads the climatological SAI  
public :: sai_finalize   ! cleanup allocated structures  
-----
```

PUBLIC TYPES:

```
-----  
public :: lis_sai         !data structure containing SAI data  
-----
```

5.17.2 sai_setup (Source File: sai_module.F90)

INTERFACE:

```
subroutine sai_setup
```

USES:

```
use lisdrv_module, only : lis
implicit none
```

ARGUMENTS:

```
integer :: n
```

DESCRIPTION:

Allocates memory for data structures for reading SAI datasets
The arguments are:

n index of the domain or nest.

5.17.3 read_sai (Source File: sai_module.F90)

INTERFACE:

```
subroutine read_sai(n)
```

USES:

```
use lisdrv_module, only : lis
implicit none
```

ARGUMENTS:

```
integer,intent(in) :: n
```

DESCRIPTION:

Reads the SAI climatology data and temporally interpolates it to the current day.
The arguments are:

n index of the domain or nest.

The routines invoked are:

readsai (5.32.4)

invokes the generic method in the registry to read the SAI climatology data

5.17.4 sai_finalize (Source File: sai_module.F90)

INTERFACE:

```
subroutine sai_finalize
```

USES:

```
use lisdrv_module, only : lis
implicit none
```

DESCRIPTION:

Deallocates objects created in this module

5.18 Fortran: Module Interface snow_module (Source File: snow_module.F90)

The code in this file implements routines to read various sources of snow depth data. Currently this module is used to ingest the SNODEP products from AFWA, which is obtained at 12z daily.

REVISION HISTORY:

08 Aug 2005: Sujay Kumar; Initial implementation

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: snow_setup      !allocates memory for required variables
public :: read_snow        !reads snow data
public :: snow_finalize    !cleanup allocated structures
```

PUBLIC TYPES:

```
public :: snow_struct      !data structure containing snow variables
```

5.18.1 snow_setup (Source File: snow_module.F90)

Allocates memory for data structures used for reading snow datasets. The snowdepth field is updated by the external files. The snow water equivalent fields are expected to be set by the model.

INTERFACE:

```
subroutine snow_setup
```

USES:

```
use lisdrv_module, only : lis
use listime_mngr, only   : setHourlyAlarm
```

5.18.2 read_snow (Source File: snow_module.F90)

This routine reads the snowdepth data at 12Z, every day.
The arguments are:

n index of the nest

INTERFACE:

```
subroutine read_snow(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use listime_mgr, only    : isHourlyAlarmRinging
```

ARGUMENTS:

```
implicit none
integer, intent(in) :: n
```

5.18.3 snow_finalize (Source File: snow_module.F90)

This routine cleans up snow-related structures

INTERFACE:

```
subroutine snow_finalize
```

USES:

```
use lisdrv_module, only : lis
```

5.19 Fortran: Module Interface soils_module (Source File: soils_module.F90)

The code in this file implements routines to read various sources of soil parameter data.

5.19.1 Overview

This module provides routines for reading and manipulating various parameters related to soil properties. The following list of soil parameters are currently supported.

sand, silt, clay fractions

soil color data

soil texture data

soil porosity data

hydraulic conductivity data

thermal conductivity data

b parameter data

quartz data

REVISION HISTORY:

```
21 Oct 2005: Sujay Kumar; Initial implementation
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
-----  
public :: soils_setup ! initializes data structures and memory  
public :: read_soils ! read soil parameters  
public :: soils_finalize !cleanup allocated structures  
-----
```

PUBLIC TYPES:

```
-----  
public :: lis_soils !data structure containing soils data  
-----
```

5.19.2 soils_setup (Source File: soils_module.F90)

INTERFACE:

```
subroutine soils_setup()
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

Allocates memory for data structures for reading soils datasets

5.19.3 read_soils (Source File: soils_module.F90)

INTERFACE:

```
subroutine read_soils()
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

Reads the soils data based on the choice of options specified in the lis configuration.
The routines invoked are:

readsoiltexture (5.37.8)

invokes the generic method in the registry to read the soil texture data

readsand (5.37.2)

invokes the generic method in the registry to read the sand fraction data

readclay (5.37.4)

invokes the generic method in the registry to read the clay fraction data

readsilt (5.37.6)

invokes the generic method in the registry to read the silt fraction data

readcolor (5.37.20)

invokes the generic method in the registry to read the soil color data

readporosity (5.37.10)

invokes the generic method in the registry to read the porosity data

readpsisat (5.37.12)

invokes the generic method in the registry to read the saturated matric potential data

readksat (5.37.14)

invokes the generic method in the registry to read the saturated hydraulic conductivity data

readbexp (5.37.16)

invokes the generic method in the registry to read the b parameter data

readquartz (5.37.18)

invokes the generic method in the registry to read the quartz data

5.19.4 soils_finalize (Source File: soils_module.F90)

INTERFACE:

```
subroutine soils_finalize()
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

deallocates memory for all datastructures used for reading soils datasets. This method is typically called after the information is translated to the LSM model tiles.

5.19.5 check_error (Source File: check_error.F90)

INTERFACE:

```
subroutine check_error(ierr,msg,iam)
```

ARGUMENTS:

```
implicit none
integer, intent(in)      :: ierr, iam
character(len=*), intent(in) :: msg
```

DESCRIPTION:

This is an error check routine. Program exits in case of error with an associated error message written to the 'abort message' file.

5.20 Fortran: Module Interface constantsmod.F90 (Source File: constantsMod.F90)

The code in this file provides values of physical constants for consistent use across different components.

REVISION HISTORY:

17 Feb 2004 Sujay Kumar Initial Specification

```
!-----  
! physical constants (all data public)  
!  
!-----  
public  
! real*8,parameter :: CONST_PI      = 3.14159265358979323846 ! pi  
real*8,parameter :: CONST_PI      = 3.14159265 ! pi  
real*8,parameter :: CONST_CDAY    = 86400.0       ! sec in calendar day ~ sec  
real*8,parameter :: CONST_SDAY    = 86164.0       ! sec in siderial day ~ sec  
real*8,parameter :: CONST_OMEGA   = 2.0*CONST_PI/CONST_SDAY ! earth rot ~ rad/sec  
real*8,parameter :: CONST_REARTH  = 6.37122e6     ! radius of earth ~ m  
real*8,parameter :: CONST_G       = 9.80616       ! acceleration of gravity ~ m/s^2  
real*8,parameter :: CONST_PSTD   = 101325.0      ! standard pressure ~ pascals  
  
real*8,parameter :: CONST_STEBOL = 5.67e-8        ! Stefan-Boltzmann constant ~ W/m^2/K^4  
real*8,parameter :: CONST_BOLTZ  = 1.38065e-23    ! Boltzmann's constant ~ J/K/molecule  
real*8,parameter :: CONST_AVOGAD = 6.02214e26     ! Avogadro's number ~ molecules/kmole  
real*8,parameter :: CONST_RGAS   = CONST_AVOGAD*CONST_BOLTZ ! Universal gas constant ~ J/K/kmole  
real*8,parameter :: CONST_MWDAIR = 28.966         ! molecular weight dry air ~ kg/kmole  
real*8,parameter :: CONST_MWWV   = 18.016         ! molecular weight water vapor  
real*8,parameter :: CONST_RDAIR  = CONST_RGAS/CONST_MWDAIR ! Dry air gas constant ~ J/K/kg  
real*8,parameter :: CONST_RWV    = CONST_RGAS/CONST_MWWV    ! Water vapor gas constant ~ J/K/kg  
real*8,parameter :: CONST_ZVIR   = (CONST_RWV/CONST_RDAIR)-1.0 ! RWV/RDAIR - 1.0  
real*8,parameter :: CONST_KARMAN = 0.4           ! Von Karman constant  
  
real*8,parameter :: CONST_TKFRZ  = 273.16        ! freezing T of fresh water ~ K (intentionally made :  
real*8,parameter :: CONST_TKTRIP = 273.16        ! triple point of fresh water ~ K  
  
real*8,parameter :: CONST_RHODAIR=CONST_PSTD/ &  
    (CONST_RDAIR*CONST_TKFRZ)          ! density of dry air at STP ~ kg/m^3  
real*8,parameter :: CONST_RHOFW  = 1.000e3        ! density of fresh water ~ kg/m^3  
real*8,parameter :: CONST_RHOSW  = 1.026e3        ! density of sea water ~ kg/m^3  
real*8,parameter :: CONST_RHOICE = 0.917e3        ! density of ice ~ kg/m^3  
real*8,parameter :: CONST_CPDAIR = 1.00464e3     ! specific heat of dry air ~ J/kg/K  
real*8,parameter :: CONST_CPFW   = 4.188e3        ! specific heat of fresh h2o ~ J/kg/K  
real*8,parameter :: CONST_CPSW   = 3.996e3        ! specific heat of sea h2o ~ J/kg/K  
real*8,parameter :: CONST_CPVW   = 1.810e3        ! specific heat of water vap ~ J/kg/K  
real*8,parameter :: CONST_CPICE  = 2.11727e3     ! specific heat of fresh ice ~ J/kg/K  
real*8,parameter :: CONST_LATICE = 3.337e5        ! latent heat of fusion ~ J/kg  
real*8,parameter :: CONST_LATVAP = 2.501e6        ! latent heat of evaporation ~ J/kg  
real*8,parameter :: CONST_LATSUB = CONST_LATICE + CONST_LATVAP ! latent heat of sublimation ~ J/kg  
  
real*8,parameter :: CONST_OCN_REF_SAL = 34.7      ! ocn ref salinity (psu)  
real*8,parameter :: CONST_ICE_REF_SAL = 4.0        ! ice ref salinity (psu)  
real*8,parameter :: CONST_SOLAR   = 1369.2        !Solar constant (W/m2)
```

5.21 Fortran: Module Interface `drv_output_mod` (Source File: `drv_output_mod.F90`)

The code in this file provides interfaces to manage LIS model output in different file formats.

5.21.1 Overview

The module provides generic interfaces to manage output from different land surface models. Currently LIS supports three data formats.

- binary: in binary, big-endian.
- grib: WMO grib1 format
- netcdf: NCAR unidata netcdf format

This module also provides generic interfaces for writing and reading model restart files. LIS uses the following convention on the filename extensions:

- .1gd4r: binary output, 1 corresponds to one variable, g represents gridded data, d represents direct access, 4r represents 4 byte real
- .grb: files in grib1 format
- .nc: files in netcdf format
- *rst: restart files. Currently all restart files are written as binary format

This module also manages any data aggregation from computing nodes when parallel processing is employed.

REVISION HISTORY:

10 Feb 2004; Sujay Kumar; Initial Specification

USES:

```
#if ( defined USE_NETCDF )
    use netcdf
#endif
    implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: drv_writevar_bin      ! write a variable into a binary file
public :: drv_writevar_grib     ! write a variable into a grib file
public :: drv_writevar_netcdf   ! write a variable into a netcdf file
public :: drv_writevar_restart ! writes a variable into a restart file
public :: drv_readvar_restart  ! reads a variable from a restart file
public :: drv_tile2grid         ! convert a tilespace variable to gridspace
```

5.21.2 `drv_writevar_bin` (Source File: `drv_output_mod.F90`)

INTERFACE:

```
interface drv_writevar_bin
```

PRIVATE MEMBER FUNCTIONS:

```
module procedure writevar_bin_int
module procedure writevar_bin_real
module procedure writevar_bin_withstats_int
module procedure writevar_bin_withstats_real
```

DESCRIPTION:

This interface provides routines for writing variables (integer or real) in a binary file. The aggregation from decomposed tile spaces in each processor to their respective grid spaces and the aggregations of these grid spaces from each processor are also performed by this routine. The interface also provides options to write some diagnostic statistics about the variable being written.

5.21.3 drv_writevar_restart (Source File: drv_output_mod.F90)

INTERFACE:

```
interface drv_writevar_restart
```

PRIVATE MEMBER FUNCTIONS:

```
module procedure writevar_restart_int
module procedure writevar_restart_real
```

DESCRIPTION:

This interface provides routines for writing variables (integer or real) in a restart file. By definition, the restart files are written in the model tile space. The aggregations from tile spaces of each processors are also performed by this routine.

5.21.4 drv_readvar_restart (Source File: drv_output_mod.F90)

INTERFACE:

```
interface drv_readvar_restart
```

PRIVATE MEMBER FUNCTIONS:

```
module procedure readvar_restart_int
module procedure readvar_restart_real
```

DESCRIPTION:

This interface provides routines for reading variables (integer or real) from a restart file. The restart files are read by the master processor. The domain decomposition of the "global" tile space on the master processor to individual processors are also performed by this routine.

5.21.5drv_writevar_grib (Source File: drv_output_mod.F90)

INTERFACE:

```
interface drv_writevar_grib
```

PRIVATE MEMBER FUNCTIONS:

```
module procedure writevar_grib_real
module procedure writevar_grib_withstats_real
```

DESCRIPTION:

This interface provides routines for writing variables (integer or real) in a grib file. The aggregation from decomposed tile spaces in each processor to their respective grid spaces and the aggregations of these grid spaces from each processor are also performed by this routine. The interface also provides options to write some diagnostic statistics about the variable being written.

5.21.6drv_writevar_netcdf (Source File: drv_output_mod.F90)

INTERFACE:

```
interface drv_writevar_netcdf
```

PRIVATE MEMBER FUNCTIONS:

```
module procedure writevar_netcdf_2d
module procedure writevar_netcdf_3d
```

DESCRIPTION:

This interface provides routines for writing variables (2d or 3d) in a netcdf file. The aggregation from decomposed tile spaces in each processor to their respective grid spaces and the aggregations of these grid spaces from each processor are also performed by this routine.

5.21.7writevar_bin_withstats_real (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call using drv_writevar_bin
subroutine writevar_bin_withstats_real(ftn,ftn_stats,n,flag,var,mvar,form)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```

integer, intent(in) :: n
integer, intent(in) :: ftn
integer, intent(in) :: ftn_stats
real, intent(in)    :: var(lis%nch(n))
character (len=*)   :: mvar
integer, intent(in) :: form
integer, intent(in) :: flag

```

DESCRIPTION:

Write a real variable to a binary output file with some diagnostic statistics written to a text file.
The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

ftn_stats unit number of the ASCII text statistics file

var variables being written, dimensioned in the tile space

mvar name of the variable being written (will be used in the stats file)

form format to be used in the stats file (1-decimal format, 2-scientific format)

flag option to determine if the variable needs to be written (1-write, 0-do not write)

The routines invoked are:

stats (5.26.4)

call to compute the diagnostic statistics

5.21.8 writevar_bin_withstats_int (Source File: **drv_output.mod.F90**)

INTERFACE:

```

Private name: cal using drv_writevar_bin
subroutine writevar_bin_withstats_int(ftn, ftn_stats, n, flag, var, mvar, form)

```

USES:

```

use lisdrv_module, only : lis,lisdom
use spmdMod

```

```

implicit none

```

ARGUMENTS:

```

integer, intent(in)    :: n
integer, intent(in)    :: ftn
integer, intent(in)    :: ftn_stats
integer, intent(in)    :: var(lis%nch(n))
character (len=*)      :: mvar
integer, intent(in)    :: form
integer, intent(in)    :: flag

```

DESCRIPTION:

Write an integer variable to a binary output file with some diagnostic statistics written to a text file.
The arguments are:

n index of the domain or nest.
ftn unit number of the binary output file
ftn_stats unit number of the ASCII text statistics file
var variables being written, dimensioned in the tile space
mvar name of the variable being written (will be used in the stats file)
form format to be used in the stats file (1-decimal format, 2-scientific format)
flag option to determine if the variable needs to be written (1-write, 0-do not write)
stats (5.26.4)
call to compute the diagnostic statistics

5.21.9 writevar_bin_real (Source File: **drv_output_mod.F90**)

INTERFACE:

```
Private name: call using drv_writevar_bin
subroutine writevar_bin_real(ftn, n, var)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn
real, intent(in)    :: var(lis%nch(n))
```

DESCRIPTION:

Write a real variable to a binary output file.

The arguments are:

n index of the domain or nest.
ftn unit number of the binary output file
var variables being written, dimensioned in the tile space

5.21.10 writevar_bin_int (Source File: **drv_output_mod.F90**)

INTERFACE:

```
Private name: call using drv_writevar_bin
subroutine writevar_bin_int(ftn, n, var)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
integer, intent(in)    :: ftn
integer, intent(in)    :: var(lis%nch(n))
```

DESCRIPTION:

Write an integer variable to a binary output file.
The arguments are:

n index of the domain or nest.
ftn unit number of the binary output file
var variables being written, dimensioned in the tile space

5.21.11 writevar_restart_real (Source File: `drv_output.mod.F90`)

INTERFACE:

```
Private name: call using drv_writevar_restart
subroutine writevar_restart_real(ftn, n, var)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: ftn
integer, intent(in) :: n
real, intent(in)   :: var(lis%nch(n))
```

DESCRIPTION:

Writes a real variable to a binary restart file.
The arguments are:

n index of the domain or nest.
ftn unit number of the binary output file
var variables being written, dimensioned in the tile space

5.21.12 writevar_restart_int (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call using drv_writevar_restart
subroutine writevar_restart_int(ftn, n, var)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: ftn
integer, intent(in) :: n
integer, intent(in) :: var(lis%nch(n))
```

DESCRIPTION:

Writes an integer variable to a binary restart file.

The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

var variables being written, dimensioned in the tile space

5.21.13 readvar_restart_real (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call drv_readvar_restart
subroutine readvar_restart_real(ftn, n, var)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: ftn
integer, intent(in)    :: n
real, intent(inout)   :: var(lis%nch(n))
```

DESCRIPTION:

Reads a real variable from a binary restart file.

The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

var variables being written, dimensioned in the tile space

5.21.14 readvar_restart_int (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call using drv_readvar_restart
subroutine readvar_restart_int(ftn, n, var)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: ftn
integer, intent(in)    :: n
integer, intent(inout) :: var(lis%ncnch(n))
```

DESCRIPTION:

Reads an integer variable from a binary restart file.

The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

var variables being written, dimensioned in the tile space

5.21.15 writevar_netcdf_2d (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call using drv_writevar_netcdf
subroutine writevar_netcdf_2d(ftn,n, var,dim1,varid)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer,intent(in) :: ftn
integer,intent(in) :: n
integer,intent(in) :: dim1
integer           :: varid
real, intent(in)   :: var(lis%glnch(n))
```

DESCRIPTION:

Writes a 2-D variable to the netcdf file

The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

dim1 index of the data write in the netcdf file (typically the time index)

varid netcdf id for the variable being written.

var variables being written, dimensioned in the tile space

5.21.16 `drv_writevar_netcdf_3d` (Source File: `drv_output.mod.F90`)

INTERFACE:

```
Private name: call using drv_writevar_netcdf
subroutine writevar_netcdf_3d(ftn,n, var, dim1, dim2, varid)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: ftn
integer, intent(in) :: dim1
integer, intent(in) :: n
integer, intent(in) :: dim2
integer, intent(in) :: varid
real, intent(in)   :: var(lis%glbnch(n))
```

DESCRIPTION:

Writes a 3-D variable to the netcdf file

The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

dim1 first index of the data write in the netcdf file (typically the time index)

dim2 second index of the data write in the netcdf file (typically the time index)

varid netcdf id for the variable being written.

var variables being written, dimensioned in the tile space

5.21.17 writevar_grib_real (Source File: drv_output_mod.F90)

INTERFACE:

```
subroutine writevar_grib_real(ftn, n, var, kpds, writeint,&
    today, yesterday)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: ftn
integer, intent(in)      :: n
real, intent(in)         :: var(lis%nch(n))
integer, intent(in)      :: kpds(25)
real, intent(in)         :: writeint
character*8,intent(in)   :: today
character*8, intent(in)  :: yesterday
```

DESCRIPTION:

Writes a real variable to a grib file

The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

var variables being written, dimensioned in the tile space

The routines invoked are:

makepdsn (5.24.5)

computes the PDS array for the variable being written

5.21.18 writevar_grib_withstats_real (Source File: drv_output_mod.F90)

INTERFACE:

```
Private name: call using drv_writevar_grib
subroutine writevar_grib_withstats_real(ftn, ftn_stats, n, var, mvar, form,&
    kpds, writeint,today, yesterday)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: ftn
integer, intent(in)      :: ftn_stats
integer, intent(in)      :: n
real, intent(in)         :: var(lis%nch(n))

character(len=*),intent(in)      :: mvar
integer, intent(in)      :: form
integer, intent(in)      :: kpds(25)
real, intent(in)         :: writeint
character*8,intent(in)    :: today
character*8,intent(in)    :: yesterday
```

DESCRIPTION:

Write a real variable to a grib output file with some diagnostic statistics written to a text file.
The arguments are:

n index of the domain or nest.

ftn unit number of the binary output file

ftn_stats unit number of the ASCII text statistics file

var variables being written, dimensioned in the tile space

mvar name of the variable being written (will be used in the stats file)

form format to be used in the stats file (1-decimal format, 2-scientific format)

The routines invoked are:

makepdsn (5.24.5)

computes the PDS array for the variable being written

stats (5.26.4)

call to compute diagnostic statistics of a variable

5.21.19 drv_tile2grid (Source File: **drv_output_mod.F90**)

INTERFACE:

```
subroutine drv_tile2grid(n,gvar,tvar)
```

USES:

```
use lisdrv_module,      only : lis,lisdom
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real               :: gvar(lis%lnc(n),lis%lnr(n))
real, intent(in)   :: tvar(lis%nch(n))
```

DESCRIPTION:

This routine converts a tile space variable to the corresponding grid space. The aggregation involves weighted average of each tile in a grid cell based on the vegetation distribution.

The arguments are:

n index of the domain or nest.

tvar variable dimensioned in the tile space.

gvar variable after conversion to the grid space

5.21.20 endrun (Source File: **endrun.F90**)

REVISION HISTORY:

14 Nov 2002 Sujay Kumar Initial Specification

INTERFACE:

subroutine endrun

USES:

```
use lis_logmod, only : lis_flush, logunit
#if (defined SPMD)
  use mpishorthand, only: MPI_COMM_WORLD
#endif
  implicit none
```

DESCRIPTION:

Routine to be called to terminate the program. This routine flushes the output streams and aborts the mpi processes.

The routines invoked are:

lis_flush (5.24.4)
flushes the output streams

5.22 Fortran: Module Interface gswp_module (Source File: **gswp_module.F90**)

This module contains useful routines that generates indices for reading the GSWP netcdf data.

REVISION HISTORY:

24 Feb 2004 Sujay Kumar Initial Specification

PUBLIC MEMBER FUNCTIONS:

```
public :: getgswp_monindex
public :: getgswp_timeindex
```

5.22.1 getgswp_monindex (Source File: gswp_module.F90)

INTERFACE:

```
subroutine getgswp_monindex(yr,mo,index)
```

ARGUMENTS:

```
integer, intent(out) :: index  
integer, intent(in) :: yr, mo
```

DESCRIPTION:

This routine computes the GSWP monthly index (base time being 1982), to be used to read monthly data.

5.22.2 getgswp_timeindex (Source File: gswp_module.F90)

INTERFACE:

```
subroutine getgswp_timeindex(yr,mo,da,hr,mn,ss,index)  
  
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: yr, mo, da, hr, mn, ss  
integer, intent(out) :: index
```

DESCRIPTION:

This routine computes the GSWP timestep index (base time being 1982), to be used to read the forcing files. Note: GSWP forcing data are written into monthly files with a frequency of 3 hours. The first entry corresponds to 03:00:00 of the first day of the given month. The last entry corresponds to 00:00:00 of the first day of the next month.

E.g.; for Tair_cru198207.nc, the data run from 1982-07-01T03:00:00 through 1982-08-01T00:00:00, inclusive. So, when you are at hour 0 on the first day of the month, reset the day to the last day of the previous month, and reset the hour from 0 to 24. This will compute the correct index for the last entry in the forcing file.

E.g.; 1982-08-01T00:00:00 should be re-written as 1982-07-31T24:00:00.

5.22.3 lapseRateCorrection (Source File: lapseRateCorrection.F90)

REVISION HISTORY:

```
11 Apr 2000: Brian Cosgrove; Initial Code  
12 May 2000: Brian Cosgrove; Corrected for zero humidities  
25 Jan 2001: Matt Rodell; Compute number of input and output  
grid points, use to allocate local arrays  
15 Mar 2001: Jon Gottschalck; if-then to handle negative vapor  
pressures in long wave correction  
14 Nov 2003: Sujay Kumar; Adopted in LIS
```

INTERFACE:

```
subroutine lapseRateCorrection(nest)
```

USES:

```
use lisdrv_module, only: lis,lisdom
use baseforcing_module, only : lisforc
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: nest
```

DESCRIPTION:

Corrects Temperature, Pressure, Humidity and Longwave Radiation forcing values for differences in elevation between the LIS running grid and the native forcing grid. The corrections are based on the lapse-rate and hypsometric adjustments to these variables described in Cosgrove et. al (2003).

Cosgrove, B.A. et.al, Real-time and retrospective forcing in the North American Land Data Assimilation (NLDAS) project, Journal of Geophysical Research, 108(D22), 8842, DOI: 10.1029/2002JD003118, 2003.

The arguments are:

nest index of the domain or nest.

5.23 Fortran: Module Interface lis_fileIOMod (Source File: lis_fileIOMod.F90)

This module contains a number of routines useful for various file I/O operations in LIS. It provides some generic routines for reading data based on the LIS run type (reading files locally or remotely), the data access method (sequential, direct access), etc. The module also provides routines to create output directories, filenames, that can be used in the model output routines.

REVISION HISTORY:

08 Apr 2004 James Geiger Initial Specification

USES:

```
use lisdrv_module, only : lis
```

PUBLIC MEMBER FUNCTIONS:

```
public :: lis_set_filename ! sets the filepaths based on the LIS type
public :: lis_open_file    ! generic file open function based on the LIS type
public :: lis_read_file    ! generic file read function based on the LIS type
public :: create_output_filename ! create an output filename
public :: create_output_directory ! create the output directory
public :: create_restart_filename ! create a restart filename
public :: create_stats_filename ! create a stats filename
public :: putget    !a generic read/write method.
```

5.23.1 putget (Source File: lis_fileIOMod.F90)

INTERFACE:

```
interface putget
```

PRIVATE MEMBER FUNCTIONS:

```
module procedure putget_int
module procedure putget_real
```

DESCRIPTION:

This is a generic method to read from or write to direct access files.

method:

- open file
- if iofunc = r, read buffer from file, abort on error
- if iofunc = w, write buffer to file, abort on error
- if iofunc = anything else, abort

REVISION HISTORY:

```
04 aug 1999 initial version .....mr gayno/dnxm
02 nov 2005 incorporated into LIS           sujay kumar
```

5.23.2 lis_set_filename (Source File: lis_fileIOMod.F90)

INTERFACE:

```
subroutine lis_set_filename(file,time_offset)
```

USES:

```
#if ( defined OPENDAP )
  use opendap_module, only : opendap_data_prefix, ciam
#endif
```

ARGUMENTS:

```
character(len=*), intent(inout):: file
character(len=*), optional :: time_offset
```

DESCRIPTION:

This routine updates the filename with the path for a GDS run. For a run using files read locally from a disk, this routine is redundant.

The arguments are:

file the updated filename

time_offset character representation of offset time

5.23.3 lis_open_file (Source File: lis_fileIOMod.F90)

INTERFACE:

```
subroutine lis_open_file(unit, file, form, status, access, recl, script, time_offset)
```

USES:

```
use lis_logmod, only : lis_log_msg
implicit none
```

ARGUMENTS:

```
integer,           intent(in) :: unit
character(len=*), intent(in) :: file
character(len=*), optional   :: form
character(len=*), optional   :: status
character(len=*), optional   :: access
integer,           optional   :: recl
character(len=*), optional   :: script
character(len=*), optional   :: time_offset
```

DESCRIPTION:

This is a generic file open routine. It parses its optional input arguments and builds an appropriate open call. It also determines whether or not data must be retrieved via a GrADS-DODS data server (GDS). If so, it calls the specified GDS script.

The arguments are:

unit unit number of the file being opened
file name of the file being opened
form format specifier - formatted or unformatted
status status of the file being accessed
access access specifier - direct or sequential
recl record length, if the file is direct access
script script being used to access the data
time_offset character representation of the offset time

The methods invoked are:

retrieve_script (5.23.6)
obtain the script used to retrieve the data
lis_log_msg (5.24.1)
write a time and processor stamped message

5.23.4 lis_read_file (Source File: lis_fileIOMod.F90)

INTERFACE:

```
subroutine lis_read_file(n, unit, array)
implicit none
```

ARGUMENTS:

```
integer,           intent(in) :: n
integer,           intent(in) :: unit
real,             intent(inout) :: array(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This routine is a generic read routine. It parses its optional input arguments and builds an appropriate read call.

The arguments are:

n index of the domain or nest
unit unit number of the file being opened
array the array being used to read the data

5.23.5 retrieve_data (Source File: lis_fileIOMod.F90)

INTERFACE:

```
subroutine retrieve_data(file, script,time_offset)
```

USES:

```
use lis_logmod, only :lis_log_msg
implicit none
```

ARGUMENTS:

```
character(len=*), intent(in) :: file
character(len=*), intent(in) :: script
character(len=*), optional :: time_offset
```

DESCRIPTION:

This routine retrieves data from a GDS. It will make 3 attempts to retrieve data from the server. If the data cannot be retrieved, this routine aborts by calling endrun.

The arguments are:

file name of the file being used
script name of the script used to retrieve data
time_offset character representation of offset time

The methods invoked are:

retrieve_script (5.23.6)
obtain the script used to retrieve the data
lis_log_msg (5.24.1)
write a time and processor stamped message
endrun (5.21.20)
call to end run in case of a fatal error

5.23.6 retrieve_script (Source File: lis_fileIOMod.F90)

INTERFACE:

```
subroutine retrieve_script(file, script, time_offset)
#if ( defined OPENDAP )
```

USES:

```
use opendap_module, only : ciam, cdom, &
                           cparm_slat, cparm_nlat, cparm_wlon, cparm_elon
use agrmetopendap_module, only : agrmet_time_index
use lis_logmod, only :lis_log_msg

character(len=*), intent(in) :: file
character(len=*), intent(in) :: script
character(len=*), optional :: time_offset
```

DESCRIPTION:

This routine retrieves makes the system call that executes the GrADS script that retrieves data from a GDS.
The arguments are:

file name of the file being used

script script being used to access the data

time_offset character representation of the offset time

The methods invoked are:

lis.log.msg (5.24.1)

write a time and processor stamped message

5.23.7 create_output_directory (Source File: lis_fileIOMod.F90)

INTERFACE:

```
subroutine create_output_directory(mname,dir_name)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only : lis_log_msg
implicit none
```

ARGUMENTS:

```
character(len=*) :: mname
character(len=40), optional :: dir_name
```

DESCRIPTION:

Create the output directory for the output data files. The call creates a hierarchy of directories in the following format, if the directory name is not specified.

joutput directory_i/EXPjexpno_i/jmodel name_i/jyr_i/jyrmoda_i

Once the directory name is created, the subroutine issues a system call to create the structure.

The arguments are:

mname a string describing the name of the model

dir_name name of the directory to override the above format

5.23.8 create_output_filename (Source File: lis_fileIOMod.F90)

INTERFACE:

```
subroutine create_output_filename(n, fname, model_name, writeint)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only : lis_log_msg

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
character(len=*), intent(out) :: fname
character(len=*), intent(in), optional :: model_name ! needed for gswp run
integer, intent(in), optional :: writeint ! output writing interval
```

DESCRIPTION:

Create the file name for the output data files. It creates both the GSWP style of output filenames and the standard LIS style. The convention used in LIS creates a filename in the following format.
joutput directoryj/EXPjexpnoj/jmodel namej/jyrj/jyrmoda{j/jyrmodahrmn{j.jextensionj
The arguments are:

n index of the domain or nest

fname the created file name.

model_name string describing the name of the model

writeint output writing interval of the model

5.23.9 create_restart_filename (Source File: lis_fileIOMod.F90)

INTERFACE:

```
subroutine create_restart_filename(n, fname, model_name, extn)
```

USES:

```
use lisdrv_module, only : lis

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
character(len=*), intent(in) :: model_name ! needed for gswp run
character(len=*), intent(out) :: fname
character(len=*), intent(in) :: extn
```

DESCRIPTION:

Create the file name for the restart data files. The convention used in LIS creates a restart filename in the following format.

|output directory|/EXP|expno|/|model name|/|yr|/|yrmnda|/|yrmodahrmn|.extn|
The arguments are:

n index of the domain or nest

fname the created file name.

model_name string describing the name of the model

extn string extension to be used in the filename

5.23.10 create_stats_filename (Source File: lis_fileIOMod.F90)

INTERFACE:

```
subroutine create_stats_filename(n, fname, mname)
```

USES:

```
use lisdrv_module, only : lis
implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: n
character(len=*), intent(out) :: fname
character(len=*), intent(in)  :: mname
```

DESCRIPTION:

Create the file name for the stats files. The convention used in LIS creates a restart filename in the following format.

|output directory|/EXP|expno|/|model name|/stats.dat
The arguments are:

n index of the domain or nest

fname the created file name.

mname string describing the name of the model

5.23.11 putget_int (Source File: lis_fileIOMod.F90)

INTERFACE:

```
!Private name: call using putget()
subroutine putget_int ( buffer, iofunc, file_name, &
                      routine_name, pgm_name, imax, jmax )
```

USES:

```
use lis_logmod, only : lis_abort
implicit none
```

ARGUMENTS:

```
integer,      intent(in)    :: imax
integer,      intent(in)    :: jmax
integer,      intent(inout)   :: buffer (imax , jmax)
character*1,   intent(in)    :: iofunc
character*100, intent(in)    :: file_name
character*10,  intent(in)    :: routine_name
character*10,  intent(in)    :: pgm_name
```

DESCRIPTION:

put or get any size integer array routine
The arguments are:

buffer integer buffer array

file_name input file name

imax grid dimension of the buffer array - East/West direction

jmax grid dimension of the buffer array - North/South direction

iofunc I/O function (r=read, w=write)

routine_name name of the calling routine

pgm_name name of the calling program

The methods invoked are:

lis_abort (5.24.2)

abort the program in case of a fatal error

5.23.12 putget_real (Source File: lis_fileIOMod.F90)

INTERFACE:

```
!Private name: call using putget()
subroutine putget_real ( buffer, iofunc, file_name, &
                       routine_name, pgm_name, imax, jmax )
```

USES:

```
use lis_logmod, only : lis_abort
implicit none

integer,      intent(in)    :: imax
integer,      intent(in)    :: jmax
real,        intent(inout)   :: buffer      (imax , jmax)
character*100, intent(in)    :: file_name
character*1,   intent(in)    :: iofunc
character*10,  intent(in)    :: routine_name
character*10,  intent(in)    :: pgm_name
```

DESCRIPTION:

put or get any size real array routine
The arguments are:

buffer integer buffer array
file_name input file name
imax grid dimension of the buffer array - East/West direction
jmax grid dimension of the buffer array - North/South direction
iofunc I/O function (r=read, w=write)
routine_name name of the calling routine
pgm_name name of the calling program

The methods invoked are:

lis_abort (5.24.2)
abort the program in case of a fatal error

5.24 Fortran: Module Interface lis_logmod (Source File: lis_logmod.F90)

The code in this file provides routines for diagnostic and error logging in LIS. A file unit number is reserved for the diagnostic log file. When a multiprocessor run is conducted, a diagnostic file from each processor is written to a separate file. An entry to the diagnostic log can be made by using:

```
write(logunit,*) msg
```

REVISION HISTORY:

12 Mar 2004: James Geiger; Initial version

PUBLIC MEMBER FUNCTIONS:

```
public :: lis_log_msg ! writes a time-processor stamped message
           ! to the diagnostic log
public :: lis_abort   ! generates a standard abort message
public :: lis_alert   ! generates a standard alert message
public :: lis_flush   ! flushes any unwritten writes to the log
```

PUBLIC TYPES:

```
public :: logunit ! file unit number used for diagnostic logging
```

5.24.1 lis_log_msg (Source File: lis_logmod.F90)

INTERFACE:

```
subroutine lis_log_msg(msg)
```

USES:

```
use spmdMod, only : iam,masterproc  
implicit none
```

ARGUMENTS:

```
character(len=*) , intent(in) :: msg
```

DESCRIPTION:

This routine formats a given message by prepending a time stamp and appending the process id number. This newly formatted message is then written to standard out.

The arguments are:

msg message to be written to the logfile

5.24.2 lis_abort (Source File: lis_logmod.F90)

REVISION HISTORY:

```
03 mar 1999 initial version .....mr moore/dnxm  
09 aug 1999 ported to IBM SP-2. increase size of abort message  
to 100 characters. removed stop 255 with a call  
to IBM utility "abort".....mr gayno/dnxm  
29 oct 2005 Sujay Kumar, Adopted in LIS
```

INTERFACE:

```
subroutine lis_abort( abort_message )  
  
implicit none  
  
character*100      :: abort_message(20)
```

DESCRIPTION:

to generate standard abort message and abort calling program.

- open abort file, write abort message and then call the system abort routine. this will cause program termination and cause script calling this routine to abort.
- if there is an error opening or writing abort file, send message to unit 6 and call the system abort routine

The arguments are:

abort_message abort message

5.24.3 lis_alert (Source File: lis_logmod.F90)

REVISION HISTORY:

```
03 mar 1999 initial version .....mr moore/dnxm
09 aug 1999 ported to IBM SP-2. increase size of message to
          100 characters. added intent attributes to
          arguments.....mr gayno/dnxm
29 oct 2005 Sujay Kumar, Adopted in LIS
```

INTERFACE:

```
subroutine lis_alert( program_name, alert_number, message )
```

5.24.4 lis_flush (Source File: lis_logmod.F90)

REVISION HISTORY:

```
16 Nov 2004 James Geiger Initial Specification
```

```
subroutine lis_flush(unit)
implicit none
integer :: unit
```

DESCRIPTION:

This routine is a generic interface to the system flush routine.
The arguments are:

unit unit to be flushed out

5.24.5 makepdsn (Source File: makepdsn.F90)

REVISION HISTORY:

```
19 Mar 2002 Cosgrove; Initial specification
```

INTERFACE:

```
subroutine makepdsn(yesterday, beforeyester, kpds, hour, writeint)
```

```
implicit none
```

ARGUMENTS:

```
character*8 :: yesterday, beforeyester
integer      :: kpds(25), hour
real         :: writeint
```

DESCRIPTION:

This routine computes the coefficients and terms required for grib output
The arguments are:

yesterday character representation of previous time
beforeyester character representation of uber previous time
kpd_s grib description array
hour current hour
writeint output writing interval in seconds

5.24.6 microMetCorrection (Source File: microMetCorrection.F90)

REVISION HISTORY:

21 Dec 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine microMetCorrection(nest)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use baseforcing_module, only : lisforc
use lis_logmod, only : logunit
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: nest
```

DESCRIPTION:

Corrects Temperature, Pressure, Humidity and Longwave Radiation values for topography (based on Liston et al; 2005) WARNING: This routine is in the priliminary testing phase.

The arguments are:

nest index of the nest

5.25 Fortran: Module Interface mpishorthand.F90 (Source File: mpishorthand.F90)

REVISION HISTORY:

02 Jan 2002 : Sujay Kumar: Initial Version

USES:

```

#if (defined SPMD)
#if (defined OSF1)
    include 'mpif.h'
#else
    use mpi
#endif
#endif

```

DESCRIPTION:

Data and parameters used for MPI. Some shorthand variables with shorter names than the standard MPI parameters. Also some variables used for heap management. Adopted from CLM

5.26 Fortran: Module Interface precision (Source File: precision.F90)

REVISION HISTORY:

14 Nov 2002; Sujay Kumar Initial Specification

ARGUMENTS:

```

integer, parameter :: r4 = selected_real_kind(5)
integer, parameter :: r8 = selected_real_kind(6)
integer, parameter :: i8 = selected_int_kind(13)

```

DESCRIPTION:

Define the precision to use for floating point and integer operations throughout the model.

5.26.1 readcard (Source File: readcard.F90)

REVISION HISTORY:

```

15 Oct 1999: Paul Houser; Initial code
23 Feb 2001: Urszula Jambor; Added GEOS or GDAS forcing option
27 Mar 2001: Jon Gottschalck; Revision of subroutine by implementing namelists
15 Apr 2002: Urszula Jambor; Added ECMWF forcing options, also
              adding 1 & 1/2 degree GLDAS domain options.
28 Apr 2002: Kristi Arsenault; Added NOAH LSM code
14 Nov 2003: Sujay Kumar; Modified card file that includes regional
              modeling options
02 Feb 2006: Sujay Kumar; Switched to the Inpak format

```

INTERFACE:

```
subroutine readcard()
```

USES:

```

use lisdrv_module, only : lis, metadata_output, config_lis
use listime_mngr, only : date2time
use param_pluginMod, only : soils_plugin, laisai_plugin,tbot_plugin,&
                           alb_plugin, gfrac_plugin, landcover_plugin,topo_plugin, &
                           snow_plugin
use runmode_pluginMod, only : runmode_plugin
use spmdMod, only :iam, masterproc
use lis_logmod, only :logunit
use LIS_ConfigMod

```

DESCRIPTION:

The code in this file initializes the LIS configuration management utility. The generic, model-independent runtime specifications of a LIS simulation are read by this routine. The routine also initializes the LIS log buffers, and calls some of the registries that set up the component plugin definitions.

The routines invoked are:

LIS_ConfigLoadFile (5.7.22)

load the specified configuration file

LIS_ConfigGetAttribute (5.7.5)

read the specified attribute

LIS_ConfigFindLabel (5.7.8)

find the specified label to read the attribute

date2time (5.4.20)

convert the date format to a floating point time format

runmode_plugin (6.1.1)

sets up function table registries for implemented runmodes

soils_plugin (6.6.2)

sets up function table registries for implemented soil data sources

laisai_plugin (6.6.3)

sets up function table registries for implemented LAI/SAI data sources

tbot_plugin (6.6.5)

sets up function table registries for implemented bottom temperature data sources

alb_plugin (6.6.7)

sets up function table registries for implemented albedo data sources

snow_plugin (6.6.8)

sets up function table registries for implemented snow depth data sources

gfrac_plugin (6.6.6)

sets up function table registries for implemented greenness fraction data sources

landcover_plugin (6.6.4)

sets up function table registries for implemented landcover data sources

topo_plugin (6.6.1)

sets up function table registries for implemented topography data sources

5.26.2 readkpds (Source File: readkpds.F90)

REVISION HISTORY:

23 Oct 2003; Sujay Kumar; Initial Version

INTERFACE:

```
subroutine readkpds(ftn, kpds, writeint)
```

```
    implicit none
```

ARGUMENTS:

```
integer :: kpds(25),ftn,writeint
```

DESCRIPTION:

Reads the kpds array from the grib table
The arguments are:

ftn the file unit number to be read from
kpds grid description array
writeint model output interval

5.26.3 soiltyp (Source File: soiltyp.F90)

3 Jul 2001: Matt Rodell Initial specification 13 Dec 2004: Sujay Kumar Updated with support for STATSGO classification

INTERFACE:

```
subroutine soiltyp(class,nc,nr,sand,clay,silt,soiltyp)
```

USES:

```
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: class
integer, intent(in) :: nc,nr
real, intent(in) :: sand(nc,nr),clay(nc,nr),silt(nc,nr)
integer, intent(inout) :: soiltyp(nc,nr)
```

DESCRIPTION:

This subroutine uses the percentages of sand, silt, and clay to convert to soil texture data. The transformation is based on the type of classification used. This routine supports the transformation to a Zobler (9 classes) or the STATSGO (19 classes) scheme.

The arguments are:

class soil classification scheme (1-zobler, 2-statsgo)
nc dimension of the grid along the east-west direction
nr dimension of the grid along the north-south direction
sand array containing the sand fraction data
clay array containing the clay fraction data
silt array containing the silt fraction data
soiltyp array containing the derived soil texture

5.26.4 stats (Source File: stats.F90)

Calculates some diagnostic statistics for a given variable for model output. The routine provides statistics such as mean, standard deviation, minimum and maximum values of a given variable.

REVISION HISTORY:

Nov 11 1999: Jon Radakovich; Initial code

INTERFACE:

```
subroutine stats(var,udef,nch,mean,stdev,min,max)
```

ARGUMENTS:

```
integer, intent(in) :: nch
real, intent(in)   :: var(nch), udef
real, intent(out)  :: mean,stdev,min,max
```

5.26.5 zterp (Source File: zterp.F90)

REVISION HISTORY:

```
10/2/98 Brian Cosgrove
6/28/00 Brian Cosgrove; changed code so that it uses LDAS%UDEF and
not a hard-wired undefined value of -999.999. Also changed
call to ZTERP subroutine so that LDAS and GRID mod files
are brought in and can be accessed in this subroutine
2/27/01 Brian Cosgrove; Added czmodel into call for ZTERP subroutine
```

INTERFACE:

```
subroutine zterp (iflag,lat,lon,btime,etime, &
mbtime,julianb,weight1,weight2,czbegdata, &
czenddata,czmodel,lis)
```

USES:

```
use lis_module
use lis_logmod, only :lis_log_msg
implicit none
```

ARGUMENTS:

```
type (lisdec) :: lis
integer       :: iflag
real          :: lat,lon
real          :: mbtime
integer       :: julianb
real          :: interval
real          :: btime
real          :: etime
real          :: weight1
real          :: weight2
```

DESCRIPTION:

This subroutine is based, in part, on modified subroutines from Jean C. Morrill of the GSWP project. The program temporally interpolates time average or instantaneous data to that needed by the model at the current timestep. It does this by combining a linear interpolation approach with a solar zenith angle approach in a fashion suitable for use with data such as short wave radiation values. It cannot be used with input data points which are more than 24 hours apart. The program outputs two weights which can then be applied to the original data to arrive at the interpolated data. If IFLAG=0, then WEIGHT1 is the weight which should be applied to the time averaged data (from the time period which the model is currently in) to arrive at the interpolated value and weight 2 is not used at all. If IFLAG=1, then WEIGHT1 should be applied to the original instantaneous data located just prior to the model time step, and WEIGHT2 should be applied to the original instantaneous data located just after the model time step.

i.e. (IF IFLAG=0) interpolated data = (WEIGHT1 * time averaged data from time period that model is currently in)

i.e. (IF IFLAG=1) interp. data = (WEIGHT1*past data)+(WEIGHT2*future data)

The arguments are:

iflag Flag specifying if input data is time averaged or not.

lis instance of the `lis_module`

lat latitude of the current point

lon longitude of the current point

btime beginning time of the original averaged data or time of original instantaneous data point which is located at or just prior to the current model time step (IFLAG=1). Expects GMT time in hour fractions. i.e., 6:30 Z would be 6.5.

etime Ending time of orig. avg. data (IFLAG=0) or time of original instantaneous data point which is located at or just after the current model time step (IFLAG=1).

mbtime Time of current time step.

julianb Julian day upon which BTIME falls

weight1 weight applied to original time averaged data (IFLAG=0) or weight applied to orig instantaneous data point located just prior to the current model time step

weight2 weight applied to orig instantaneous data point located just after the current model time step (IFLAG=1) If IFLAG=0, then this weight is meaningless and should not be used

czbegdata cosine of zenith angle at the beginning timestep

czenddata cosine of zenith angle at the ending timestep

czmodel cosine of the zenith angle at the current timestep

The routines invoked are:

localtime (5.26.7)

compute the localtime based on the GMT

coszenith (5.26.6)

compute the cosine of zenith angle

5.26.6 coszenith (Source File: zterp.F90)

Returns the cosine of the zenith angle from the following parameters

- 1) Day angle (GAMMA)
- 2) Solar DEClination
- 3) Equation of time
- 4) Local apparent time
- 5) Hour angle
- 6) Cosine of zenith angle

All equations come from "An Introduction to Solar Radition" By Muhammad Iqbal, 1983.

INTERFACE:

```
subroutine coszenith (lon,latd,lhour,zone,julian,czenith,dec,omega)
```

```
    implicit none
```

ARGUMENTS:

integer :: zone	! time zone (1-24) gmt=12
integer :: julian	! julian day
real :: czenith	! cosine of zenith angle (radians)
real :: dec	! solar declination (radians)
real :: et	! equation of time (minutes)
real :: gamma	! day angle (radians)
real :: latime	! local apparent time
real :: lcorr	! longitudical correction
real :: lhour	! local standard time
real :: lon	! local longitude (deg)
real :: llat	! local latitude in radians
real :: latd	! local latitude in degrees
real :: ls	! standard longitude (deg)
real :: omegad	! omega in degrees
real :: omega	! omega in radians
real :: pi	! universal constant pi [-]
real :: zenith	! zenith angle(radians)

5.26.7 localtime (Source File: zterp.F90)

Calculates the local time based on GMT

INTERFACE:

```
subroutine localtime (gmt,lon,lhour,zone)
```

ARGUMENTS:

real:: gmt	! GMT time (0-23)
real:: lon	! longitude in degrees
real:: change	! the change in number of hours between
real:: lhour	! local hour (0-23) 0= midnight, 23= 11:00 p.m.
integer:: i	! working integer
integer:: zone	! time zone (1-24)

5.27 Fortran: Module Interface LIS_alb_FTable (Source File: LIS_alb_FTable.c)

Function table registries for storing the interface implementations for managing different sources of albedo parameter data

5.27.1 registerreadmxsnoalb (Source File: LIS_alb_FTable.c)

INTERFACE:

```
void FTN(registerreadmxsnoalb)(int *i, int *j,void (*func)(int*, float*))
```

DESCRIPTION:

Creates a entry in the registry for the routine to open and read albedo upper bound over deep snow
The arguments are:

i index of the domain

j index of the albedo source

5.27.2 readmxsnoalb (Source File: LIS_alb_FTable.c)

INTERFACE:

```
void FTN(readmxsnoalb)(int *n, int *i, int *j,float *array)
```

DESCRIPTION:

Invokes the routines from the registry for reading albedo upper bound over deep snow data
The arguments are:

i index of the domain

j index of the albedo source

5.27.3 registerreadalbedo (Source File: LIS_alb_FTable.c)

INTERFACE:

```
void FTN(registerreadalbedo)(int *i, int *j,void (*func)(int*, int*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read albedo climatology data
The arguments are:

i index of the domain

j index of the albedo source

5.27.4 readalbedo (Source File: LIS_alb_FTable.c)

INTERFACE:

```
void FTN(readalbedo)(int *n, int *i, int *j, int *k, float *array)
```

DESCRIPTION:

Invokes the routines from the registry for reading climatology albedo files
The arguments are:

i index of the domain

j index of the albedo source

5.28 Fortran: Module Interface LIS_dataassim_FTable (Source File: LIS_dataassim_FTable.c)

Function table registries for storing the interface implementations for the operation of various data assimilation operations.

5.28.1 registerdaobssetup (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(registerdaobssetup)(int *i, void (*func)(int*))
```

DESCRIPTION:

Creates an entry in the registry for the routine that sets up structures for handling observation data for data assimilation

The arguments are:

i index of the observation data source

5.28.2 dataobssetup (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(dataobssetup)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to set up structures for handling observation data for data assimilation
The arguments are:

i index of the observation data source

5.28.3 registerreadobs (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(registerdaobs)(int *i, void (*func)(int*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to reading observations for data assimilation
The arguments are:

i index of the observation data source

5.28.4 readobservations (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(readobservations)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to read observations for data assimilation
The arguments are:

i index of the observation data source

5.28.5 registerdasetup (Source File: LIS_dataassim_FTable.c)

Creates an entry in the registry for the routine to set up algorithm specific structures for the data assimilation method used.

The arguments are:

i index of the assimilation algorithm

j index of of the variable being assimilated

INTERFACE:

```
void FTN(registerdasetup)(int *i, int *j, void (*func)(int*))
```

5.28.6 dataassimsetup (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(dataassimsetup)(int *i, int *j)
```

DESCRIPTION:

Invokes the routine from the registry to set up the algorithm specific structures for the data assimilation algorithm used.

The arguments are:

i index of the assimilation algorithm
j index of of the variable being assimilated

5.28.7 registerfrcst (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(registerfrcst)(int *i, int *j, void (*func)(int*))
```

DESCRIPTION:

Creates and entry in the registry for the routine that performs the forecast step
The arguments are:

i index of the assimilation algorithm
j index of of the variable being assimilated

5.28.8 forecast (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(forecast)(int *i,int *j)
```

DESCRIPTION:

Invokes the routine from the registry that performs the forecast step
The arguments are:

i index of the assimilation algorithm
j index of of the variable being assimilated

5.28.9 registerassim (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(registerassim)(int *i, int *j, void (*func)(int*))
```

DESCRIPTION:

Creates an entry in the registry for the routine that performs the assimilation step
The arguments are:

i index of the assimilation algorithm
j index of of the variable being assimilated

5.28.10 assimilate (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(assimilate)(int *i,int *j)
```

DESCRIPTION:

Invoke the routine from the registry that performs the the data assimilation step
The arguments are:

i index of the assimilation algorithm

j index of of the variable being assimilated

5.28.11 registerdafinalize (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(registerdafinalize)(int *i, void (*func)(int*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to cleanup allocated structures
The arguments are:

i index of the assimilation algorithm

j index of of the variable being assimilated

5.28.12 dafinalize (Source File: LIS_dataassim_FTable.c)

INTERFACE:

```
void FTN(dafinalize)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry that cleans up the data assimilation specific structures
The arguments are:

i index of the assimilation algorithm

5.29 Fortran: Module Interface LIS_domain_FTable (Source File: LIS_domain_FTable.c)

Function table registries for storing the interface implementations for managing the operation of different domain specifications

5.29.1 registerdomain (Source File: LIS_domain_FTable.c)

INTERFACE:

```
void FTN(registerdomain)(int *i,void (*func)())
```

DESCRIPTION:

Creates an entry in the registry for the routine to create grid and tile spaces for a particular domain definition
The arguments are:

i index of the domain

5.29.2 makedomain (Source File: LIS_domain_FTable.c)

INTERFACE:

```
void FTN(makedomain)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to create grid and tile spaces for a particular domain definition
The arguments are:

i index of the domain

5.29.3 registerinput (Source File: LIS_domain_FTable.c)

INTERFACE:

```
void FTN(registerinput)(int *i,void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to read the runtime domain specifics
The arguments are:

i index of the domain

5.29.4 readinput (Source File: LIS_domain_FTable.c)

INTERFACE:

```
void FTN(readinput)(int *i)
```

DESCRIPTION:

Calls the routine from the registry to read the the runtime domain specifics
The arguments are:

i index of the domain

5.30 Fortran: Module Interface LIS_forcing_FTable (Source File: LIS_forcing_FTable.c)

Function table registries for storing the interface implementations for managing different meteorological analyses implemented as "baseforcings"

5.30.1 registerget (Source File: LIS_forcing_FTable.c)

INTERFACE:

```
void FTN(registerget)(int *i,void (*func)(int*))
```

DESCRIPTION:

Creates an entry in the registry for the routines to open and read base forcing
The arguments are:

i index of the forcing type

5.30.2 retrieverforcing (Source File: LIS_forcing_FTable.c)

INTERFACE:

```
void FTN(retrieverforcing)(int *i, int *n)
```

DESCRIPTION:

Invokes the routine from the registry to open and read the base forcing
The arguments are:

i index of the forcing type

n index of the nest

5.30.3 registertimeinterp (Source File: LIS_forcing_FTable.c)

INTERFACE:

```
void FTN(registertimeinterp)(int *i,void (*func)(int*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to perform temporal interpolation
The arguments are:

i index of the forcing type

5.30.4 timeinterp (Source File: LIS_forcing_FTable.c)

INTERFACE:

```
void FTN(timeinterp)(int *i, int *n)
```

DESCRIPTION:

Invokes the routine from the registry to perform temporal interpolation
The arguments are:

i index of the forcing type

n index of the nest

5.30.5 registerdefinenative (Source File: LIS_forcing_FTable.c)

INTERFACE:

```
void FTN(registerdefinenative)(int *i, void (*func)())
```

DESCRIPTION:

Creates an entry in the registry for the routine that define the native domain of the base forcing scheme.
The arguments are:

i index of the forcing type

5.30.6 definenative (Source File: LIS_forcing_FTable.c)

INTERFACE:

```
void FTN(definenative)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry that defines the native domain of the base forcine scheme.
The arguments are:

i index of the forcing type

5.30.7 registerforcingfinal (Source File: LIS_forcing_FTable.c)

INTERFACE:

```
void FTN(registerforcingfinal)(int *i, void (*func)())
```

DESCRIPTION:

Creates an entry in the registry for the routine to cleanup allocated structures
The arguments are:

i index of the forcing type

5.30.8 forcingfinalize (Source File: LIS_forcing_FTable.c)

INTERFACE:

```
void FTN(forcingfinalize)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to cleanup the allocated structures associated with the base forcing
The arguments are:

i index of the forcing type

5.31 Fortran: Module Interface LIS_gfrac_FTable (Source File: LIS_gfrac_FTable.c)

Function table registries for storing the interface implementations for managing different sources of greenness fraction data

5.31.1 registerreadgfrac (Source File: LIS_gfrac_FTable.c)

INTERFACE:

```
void FTN(registerreadgfrac)(int *i,int *j,void (*func)(int*, int*, float*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to read gfrac data
The arguments are:

i index of the domain

j index of the greenness data source

5.31.2 readgfrac (Source File: LIS_gfrac_FTable.c)

INTERFACE:

```
void FTN(readgfrac)(int *n, int *i,int *j,int *month, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to reading gfrac data
The arguments are:

n index of the nest

i index of the domain

j index of the greenness data source

month month

array pointer to the greenness data

5.32 Fortran: Module Interface LIS_laisai_FTable (Source File: LIS_laisai_FTable.c)

Function table registries for storing the interface implementations for managing different sources of LAI and SAI data

5.32.1 registerreadlai (Source File: LIS_laisai_FTable.c)

INTERFACE:

```
void FTN(registerreadlai)(int *i, int *j, void (*func)(int*, float*,float*,float*,float*))
```

DESCRIPTION:

Creates an entry in the registry for the routines to read LAI data
The arguments are:

i index of domain

j index of the lai source

5.32.2 readlai (Source File: LIS_laisai_FTable.c)

INTERFACE:

```
void FTN(readlai)(int *n, int *i, int *j, float *array1, float *array2, float* wt1, float* wt2)
```

DESCRIPTION:

Invokes the routine from the registry to read LAI data.
The arguments are:

i index of domain

j index of the LAI source

n index of the nest

array1, array2 pointers to LAI data

wt1, wt2 time interpolation weights

5.32.3 registerreadsai (Source File: LIS_laisai_FTable.c)

INTERFACE:

```
void FTN(registerreadsai)(int *i,int *j,void (*func)(int*, float*,float*,float*,float*))
```

DESCRIPTION:

Creates an entry in the registry for the routines to read SAI data

The arguments are:

i index of domain

j index of the LAI source

5.32.4 readsai (Source File: LIS_laisai_FTable.c)

INTERFACE:

```
void FTN(readsai)(int *n, int *i, int *j, float *array1, float *array2, float* wt1, float* wt2)
```

DESCRIPTION:

Invokes the routine from the registry to read SAI data.

The arguments are:

i index of domain

j index of the LAI source

n index of the nest

array1, array2 pointers to LAI data

wt1, wt2 time interpolation weights

5.33 Fortran: Module Interface LIS_landcover_FTable (Source File: LIS_landcover_FTable.c)

Function table registries for storing the interface implementations for managing different sources of landcover data

5.33.1 registerreadlc (Source File: LIS_landcover_FTable.c)

INTERFACE:

```
void FTN(registerreadlc)(int *i,int *j, void (*func)())
```

DESCRIPTION:

Creates an entry in the registry for the routine to read landcover data

The arguments are:

i index of the domain

j index of the landcover source

5.33.2 readlandcover (Source File: LIS_landcover_FTable.c)

Invokes the routine from the registry for reading landcover data.

The arguments are:

n index of the nest

i index of the domain

j index of the landcover source

array pointer to the landcover data

INTERFACE:

```
void FTN(readlandcover)(int *n, int *i, int *j, float *array)
```

5.33.3 registerreadmask (Source File: LIS_landcover_FTable.c)

INTERFACE:

```
void FTN(registerreadmask)(int *i, int *j, void (*func)())
```

DESCRIPTION:

Creates an entry in the registry for the routine to read landmask data
The arguments are:

i index of the domain

j index of the landcover source

5.33.4 readlandmask (Source File: LIS_landcover_FTable.c)

INTERFACE:

```
void FTN(readlandmask)(int *n, int *i, int *j, float *array)
```

DESCRIPTION:

Invokes the routine from the registry for reading landmask data.
The arguments are:

n index of the nest

i index of the domain

j index of the landcover source

array pointer to the landmask data

5.34 Fortran: Module Interface LIS_lsm_FTable (Source File: LIS_lsm_FTable.c)

Function table registries for storing the interface implementations for managing the operations of different land surface models

5.34.1 registerlsmini (Source File: LIS_lsm_FTable.c)

Creates an entry in the registry for the routine to perform land surface model initialization

INTERFACE:

```
void FTN(registerlsmini)(int *i, void (*func)())
```

5.34.2 lsmini (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmini)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to perform land surface model initialization

i index of the LSM

5.34.3 registerlsmrun (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmrun)(int *i, void (*func)(int*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to run the land surface model

i index of the LSM

5.34.4 lsmrun (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmrun)(int *i,int *n)
```

DESCRIPTION:

Invokes the routine from the registry to run the land surface model

i index of the LSM

n index of the nest

5.34.5 registerlsmfinalize (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmfinalize)(int *i, void (*func)())
```

DESCRIPTION:

Creates an entry in the registry for the routine to cleanup allocated structures specific to the land surface model

i index of the LSM

5.34.6 lsmfinalize (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmfinalize)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry for cleaning up allocated structures specific to the land surface model

i index of the LSM

5.34.7 registerlsmsetup (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmsetup)(int *i, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to set up land surface model parameters

i index of the LSM

5.34.8 lsmsetup (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmsetup)(int *i)
```

DESCRIPTION:

Invokes the routine in the registry to set up land surface model parameters

i index of the LSM

5.34.9 registerlsmrestart (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmrestart)(int *i, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to restart the land surface model from a previously saved state
i index of the LSM

5.34.10 lsmrestart (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmrestart)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to restart the land surface model from a previously saved state
i index of the LSM

5.34.11 registerlsmoutput (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmoutput)(int *i, void (*func)(int*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to perform land surface model output
i index of the LSM

5.34.12 lsmoutput (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmoutput)(int *i, int *n)
```

DESCRIPTION:

Invokes the routine from the registry to perform land surface model output
i index of the LSM
n index of the nest

5.34.13 registerlsmf2t (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmf2t)(int *i, void (*func)(int*, int*, float*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to transfer forcing to model tiles

i index of the LSM

5.34.14 lsmf2t (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmf2t)(int *i, int*n, int *index, float *forcing)
```

DESCRIPTION:

Invokes the routine from the registry to transferr forcing to model tiles

i index of the LSM

n index of the nest

index index of the tile

forcing pointer to the forcing array

5.34.15 registerlsmwrst (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmwrst)(int *i, void (*func)(int*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to write restart files for a land surface model

i index of the LSM

5.34.16 lsmwrst (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmwrst)(int *i, int *n)
```

DESCRIPTION:

Invokes the routine from the registry to write restart files for a land surface model

i index of the LSM

n index of the nest

5.34.17 registergetstatevar (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registergetstatevar)(int *i, int *j, void (*func)(float*))
```

DESCRIPTION:

Makes an entry in the registry for the routine for obtaining the specified state variable from the land surface model

i index of the LSM

5.34.18 getstatevar (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(getstatevar)(int *i, int *j, float *statevars)
```

DESCRIPTION:

Invokes the routine from the registry for obtaining the specified state variable from the land surface model

i index of the LSM

j index of the variable

statevars pointer to the state variable array

5.34.19 registersetStatevar (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registersetStatevar)(int *i, int *j, void (*func)(float*))
```

DESCRIPTION:

Makes an entry in the registry for updating the specified state variable in a land surface model

i index of the LSM

j index of the variable

5.34.20 setStatevar (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(setStatevar)(int *i, int *j, float *statevars)
```

DESCRIPTION:

Invokes the routine from the registry for updating the specified state variable in a land surface model

i index of the LSM

j index of the variable

statevars pointer to the state variable array

5.34.21 registerGetNPr (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerGetNPr)(int *i, void (*func)(int*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to obtain the number of prognostic variables to be used data assimilation

i index of the LSM

5.34.22 getnpr (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(getnpr)(int *i, int *npr)
```

DESCRIPTION:

Invokes the routine from the registry to obtain the number of prognostic variables used in data assimilation

i index of the LSM

npr number of prognostic variables

5.34.23 registerlsmreset (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmreset)(int *i, void (*func)(int*))
```

DESCRIPTION:

Makes an entry in the registry to reset the LSM back to a previous state

i index of the LSM

5.34.24 lsmreset (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmreset)(int *i, int *flag)
```

DESCRIPTION:

Invokes the routine from the registry to reset the LSM to a previous state

i index of the LSM

flag flag for reset option

5.34.25 registerobstransform (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerobstransform)(int *i, int *j, void (*func)(float*, float*))
```

DESCRIPTION:

Makes an entry in the registry to perform the translation of observations to state variables

i index of the LSM

j index of the observation data

5.34.26 obstransform (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(obstransform)(int *i, int *j, float *vsm, float *obsst)
```

DESCRIPTION:

Invokes the routine from the registry to translate the observations to state variables

i index of the LSM

j index of the observation data

vsm variable being transformed

obsst transformed variable

5.34.27 registerlsmexport (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmexport)(int *i, void (*func)(int*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to set export states for a LSM

i index of the LSM

5.34.28 lsmsetexport (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmsetexport)(int *i, int *n)
```

DESCRIPTION:

Invokes the routine in the registry to set the export states for a LSM

i index of the LSM

n index of the nest

5.34.29 registerlsmdynsetup (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(registerlsmdynsetup)(int *i, void (*func)(int*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to set the time dependent land surface parameters

i index of the LSM

5.34.30 lsmdynsetup (Source File: LIS_lsm_FTable.c)

INTERFACE:

```
void FTN(lsmdynsetup)(int *i, int *n)
```

DESCRIPTION:

Invokes the routine from the registry to set the time dependent land surface parameters

i index of the LSM

n index of the nest

5.35 Fortran: Module Interface LIS_perturb_FTable (Source File: LIS_perturb_FTable.c)

Function table registries for storing the interface implementations for managing the operations of different perturbation algorithms

5.35.1 registerperturbsetup (Source File: LIS_perturb_FTable.c)

INTERFACE:

```
void FTN(registerperturbinit)(int *i, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to initialize the perturbation scheme.
The arguments are:

i index of the perturbation algorithm .

5.35.2 perturbinit (Source File: LIS_perturb_FTable.c)

INTERFACE:

```
void FTN(perturbinit)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to initialize the perturbation scheme

The arguments are:

i index of the perturbation algorithm .

5.35.3 registerperturb (Source File: LIS_perturb_FTable.c)

INTERFACE:

```
void FTN(registerperturb)(int *i, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to call the perturbation algorithm.

The arguments are:

i index of the perturbation algorithm .

5.35.4 perturb (Source File: LIS_perturb_FTable.c)

INTERFACE:

```
void FTN(perturb)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to call the perturbation algorithm.

The arguments are:

i index of the perturbation algorithm .

5.35.5 registergetpertforcing (Source File: LIS_perturb_FTable.c)

INTERFACE:

```
void FTN(registergetpertforcing)(int *i, void (*func)(int*, int*, int*, float*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to retrieve the perturbed forcing

The arguments are:

i index of the perturbation algorithm .

5.35.6 getpertforcing (Source File: LIS_perturb_FTable.c)

INTERFACE:

```
void FTN(getpertforcing)(int *i, int *c, int *r, int *m, float *forc1, float *forc2)
```

DESCRIPTION:

Invokes the routine from the registry to retrieve the perturbed forcing
The arguments are:

i index of the perturbation algorithm

c index of the column

r index of the row

forc1 input forcing

forc2 output perturbed forcing .

5.36 Fortran: Module Interface LIS_runmode_FTable (Source File: LIS_runmode_FTable.c)

Function table registries for storing the interface implementations of different running modes in LIS

5.36.1 registerlisinit (Source File: LIS_runmode_FTable.c)

INTERFACE:

```
void FTN(registerlisinit)(int *i, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the LIS initialization method for a certain running mode
The arguments are:

i index of the running mode

5.36.2 lisinit (Source File: LIS_runmode_FTable.c)

INTERFACE:

```
void FTN(lisinit)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to perform the LIS initialization for the specified running mode
The arguments are:

i index of the running mode

5.36.3 registerlisrun (Source File: LIS_runmode_FTable.c)

INTERFACE:

```
void FTN(registerlisrun)(int *i, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the LIS run method for a certain running mode.
The arguments are:

i index of the running mode

5.36.4 lisrun (Source File: LIS_runmode_FTable.c)

INTERFACE:

```
void FTN(lisrun)(int *i)
```

DESCRIPTION:

Invokes the LIS run method from the registry for the specified running mode
The arguments are:

i index of the running mode

5.36.5 registerlisfinalize (Source File: LIS_runmode_FTable.c)

INTERFACE:

```
void FTN(registerlisfinalize)(int *i, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to perform LIS finalization for the specified running mode
The arguments are:

i index of the running mode

5.36.6 lisfinalize (Source File: LIS_runmode_FTable.c)

INTERFACE:

```
void FTN(lisfinalize)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to perform LIS finalization call for the specified running mode
The arguments are:

i index of the running mode

5.37 Fortran: Module Interface LIS_soils_FTable (Source File: LIS_soils_FTable.c)

Function table registries for storing the interface implementations for managing different sources of soil parameters data

5.37.1 registerreadsand (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(registerreadsand)(int *i,int *j, void (*func)(int *, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read the sand fraction data
The arguments are:

i index of the domain

j index of the soils source

5.37.2 readsand (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readsand)(int *n, int *i, int *j,float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the sand fraction data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the sand array

5.37.3 registerreadclay (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(registerreadclay)(int *i, int *j,void (*func)(int*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read the clay fraction data
The arguments are:

i index of the domain

j index of the soils source

5.37.4 readclay (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readclay)(int *n, int *i, int *j, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the clay fraction data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the clay data

5.37.5 registerreadsilt (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(registerreadsilt)(int *i, int *j, void (*func)(int*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read the silt fraction data
The arguments are:

i index of the domain

j index of the soils source

5.37.6 readsilt (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readsilt)(int *n, int *i, int *j, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the silt fraction data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the silt data

5.37.7 registerreadtexture (Source File: LIS_soils_FTable.c)

Creates an entry in the registry for the routine to read the soil texture data
The arguments are:

i index of the domain

j index of the soils source

INTERFACE:

```
void FTN(registerreadsoiltexture)(int *i, int *j,void (*func)(int*, int*))
```

5.37.8 readsoiltexture (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readsoiltexture)(int *n, int *i, int *j,int *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the soil texture data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the texture array

5.37.9 registerreadporosity (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(registerreadporosity)(int *i, int *j,void (*func)(int*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read the porosity data
The arguments are:

i index of the domain

j index of the soils source

5.37.10 readporosity (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readporosity)(int *n, int *i, int *j, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the porosity data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the porosity array

5.37.11 registerreadpsisat (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(registerreadpsisat)(int *i, int *j, void (*func)(int*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read the saturated matric potential data
The arguments are:

i index of the domain

j index of the soils source

5.37.12 readpsisat (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readpsisat)(int *n, int *i, int *j, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the saturated matric potential data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the saturated matric potential data

5.37.13 registerreadksat (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(registerreadksat)(int *i, int *j,void (*func)(int*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read the saturated hydraulic conductivity data
The arguments are:

i index of the domain

j index of the soils source

5.37.14 readksat (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readksat)(int *n, int *i, int *j,float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the saturated hydraulic conductivity data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the ksat data

5.37.15 registerreadbexp (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(registerreadbexp)(int *i, int *j,void (*func)(int*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read the b parameter data
The arguments are:

i index of the domain

j index of the soils source

5.37.16 readbexp (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readbexp)(int *n, int *i, int *j, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the b parameter data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the b parameter data

5.37.17 registerreadquartz (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(registerreadquartz)(int *i, int *j, void (*func)(int*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read the quartz fraction data
The arguments are:

i index of the domain

j index of the soils source

5.37.18 readquartz (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readquartz)(int *n, int *i, int *j, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the quartz fraction data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the quartz data

5.37.19 registerreadcolor (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(registerreadcolor)(int *i,int *j, void (*func)(int*, float*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to read the soil color data
The arguments are:

i index of the domain

j index of the soils source

5.37.20 readcolor (Source File: LIS_soils_FTable.c)

INTERFACE:

```
void FTN(readcolor)(int *n, int *i, int *j,float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the soil color data
The arguments are:

n index of the nest

i index of the domain

j index of the soils source

array pointer to the color array

5.38 Fortran: Module Interface LIS_suppforcing_FTable (Source File: LIS_suppforcing_FTable.f90)

Function table registries for storing the interface implementations for managing different supplemental forcing analyses

5.38.1 registerdefinenativesupp (Source File: LIS_suppforcing_FTable.c)

INTERFACE:

```
void FTN(registerdefinenativesupp)(int *i, void (*func)())
```

DESCRIPTION:

Creates an entry in the registry for the routine that define the native domain of the supplemental forcing scheme.

The arguments are:

i index of the forcing type

5.38.2 definenativesupp (Source File: LIS_suppforcing_FTable.c)

INTERFACE:

```
void FTN(definenativesupp)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry that defines the native domain of the supplemental forcing scheme.
The arguments are:

i index of the forcing type

5.38.3 registerreadsupp (Source File: LIS_suppforcing_FTable.c)

INTERFACE:

```
void FTN(registerreadsupp)(int *i, void (*func)(int*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to open and read the supplemental forcing
The arguments are:

i index of the forcing type

5.38.4 getsuppforc (Source File: LIS_suppforcing_FTable.c)

INTERFACE:

```
void FTN(getsuppforc)(int *n, int *i)
```

DESCRIPTION:

Invokes the routine from the registry to open and read the supplemental forcing
The arguments are:

n index of the nest

i index of the forcing type

5.38.5 registersuppti (Source File: LIS_suppforcing_FTable.c)

INTERFACE:

```
void FTN(registeruppti)(int *i, void (*func)(int*))
```

DESCRIPTION:

Creates an entry in the registry for the routine to perform temporal interpolation of the supplemental forcing
The arguments are:

i index of the forcing type

5.38.6 timeinterpsupp (Source File: LIS_suppforcing_FTable.c)

INTERFACE:

```
void FTN(timeinterpsupp)(int *n, int *i)
```

DESCRIPTION:

Invokes the routine from the registry to perform temporal interpolation of the supplemental forcing
The arguments are:

n index of the nest

i index of the forcing type

5.38.7 registersuppforcingfinal (Source File: LIS_suppforcing_FTable.c)

INTERFACE:

```
void FTN(registersuppforcingfinal)(int *i, void (*func)())
```

DESCRIPTION:

Creates an entry in the registry for the routine to cleanup allocated structures
The arguments are:

i index of the forcing type

5.38.8 suppforcingfinalize (Source File: LIS_suppforcing_FTable.c)

INTERFACE:

```
void FTN(suppforcingfinalize)(int *i)
```

DESCRIPTION:

Invokes the routine from the registry to cleanup the allocated structures associated with the supplemental forcing

The arguments are:

i index of the forcing type

5.39 Fortran: Module Interface LIS_tbot_FTable (Source File: LIS_tbot_FTable.c)

Function table registries for storing the interface implementations for managing different sources of bottom temperature data

5.39.1 registerreadtbot (Source File: LIS_tbot_FTable.c)

INTERFACE:

```
void FTN(registerreadtbot)(int *i, void (*func)(int*, float*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to read the bottom temperature data
The arguments are:

i index of the domain

5.39.2 readtbot (Source File: LIS_tbot_FTable.c)

INTERFACE:

```
void FTN(readtbot)(int *n, int *i, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the bottom temperature data
The arguments are:

i index of the domain

5.40 Fortran: Module Interface LIS_topo_FTable (Source File: LIS_topo_FTable.c)

Function table registries for storing the interface implementations for managing different sources of topography data

5.40.1 registerreadelev (Source File: LIS_topo_FTable.c)

INTERFACE:

```
void FTN(registerreadelev)(int *i, int *j, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to read elevation data
The arguments are:

i index of the domain

j index of the topography source

5.40.2 readelev (Source File: LIS_topo_FTable.c)

INTERFACE:

```
void FTN(readelev)(int *n,int *i, int *j, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the elevation data
The arguments are:

n index of the nest

i index of the domain

j index of the topography source

array pointer to the elevation array

5.40.3 registerreadslope (Source File: LIS_topo_FTable.c)

INTERFACE:

```
void FTN(registerreadslope)(int *i,int *j, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to read slope data
The arguments are:

i index of the domain

j index of the topography source

5.40.4 readslope (Source File: LIS_topo_FTable.c)

INTERFACE:

```
void FTN(readslope)(int *n,int *i, int *j, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the slope data
The arguments are:

n index of the nest

i index of the domain

j index of the topography source

array pointer to the slope array

5.40.5 registerreadaspect (Source File: LIS_topo_FTable.c)

INTERFACE:

```
void FTN(registerreadaspect)(int *i,int*j, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to read aspect data
The arguments are:

i index of the domain

j index of the topography source

5.40.6 readaspect (Source File: LIS_topo_FTable.c)

INTERFACE:

```
void FTN(readaspect)(int *n,int *i, int *j, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the aspect data
The arguments are:

n index of the nest

i index of the domain

j index of the topography source

array pointer to the aspect array

5.40.7 registerreadcurv (Source File: LIS_topo_FTable.c)

INTERFACE:

```
void FTN(registerreadcurv)(int *i,int *j, void (*func)())
```

DESCRIPTION:

Makes an entry in the registry for the routine to read curvature data
The arguments are:

i index of the domain

j index of the topography source

5.40.8 readcurv (Source File: LIS_topo_FTable.c)

INTERFACE:

```
void FTN(readcurv)(int *n,int *i, int *j, float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read the curvature data

The arguments are:

n index of the nest

i index of the domain

j index of the topography source

array pointer to the curvature array

5.41 Fortran: Module Interface LIS_snowd_FTable (Source File: LIS_snowd_FTable.c)

Function table registries for storing the interface implementations for managing different sources of snow depth data.

5.41.1 registerreadsnowdepth (Source File: LIS_snowd_FTable.c)

INTERFACE:

```
void FTN(registerreadsnowdepth)(int *i,int *j,void (*func)(int*, float*))
```

DESCRIPTION:

Makes an entry in the registry for the routine to read snow depth data.

The arguments are:

i index of the domain

j index of the snow depth data source

5.41.2 readsnowdepth (Source File: LIS_snowd_FTable.c)

INTERFACE:

```
void FTN(readsnowdepth)(int *n, int *i,int *j,float *array)
```

DESCRIPTION:

Invokes the routine from the registry to read snow depth data

The arguments are:

n index of the nest

i index of the domain

j index of the snow depth data source

array pointer to the snow depth data

5.41.3 lis_log_msgC.c (Source File: lis_log_msgC.c)

This file provides methods for diagnostic and error logging in LIS for codes written in the C language. The diagnostic messages are appended with the appropriate time-date and processor stamps.

5.41.4 lis_log_msgC (Source File: lis_log_msgC.c)

This routine is a C wrapper to the lis_log_msg routine.

REVISION HISTORY:

12 Mar 2004: James Geiger; Initial version

INTERFACE:

```
void lis_log_msgC(char * string)
```

5.41.5 lis_memory_managementC (Source File: lis_memory_managementC.c)

This file provides methods for dynamic memory management in LIS for codes written in the C language.

5.41.6 lis_calloc (Source File: lis_memory_managementC.c)

Generic call to the C calloc function used in LIS.

REVISION HISTORY:

Mar 2004, James Geiger; Initial Specification

INTERFACE:

```
void * lis_calloc(size_t n, size_t size, char * caller)
```

5.41.7 lis_malloc (Source File: lis_memory_managementC.c)

Generic call to the C malloc function used in LIS.

REVISION HISTORY:

Mar 2004, James Geiger; Initial Specification

INTERFACE:

```
void * lis_malloc(size_t size, char * caller)
```

5.41.8 listask_for_point (Source File: listask_for_point.c)

This file provides an algorithm for the decomposition of a given domain. The method is adopted from the Weather Research and Forecasting (WRF) source code.

REVISION HISTORY:

Feb 2006 Sujay Kumar Adopted in LIS

Part IV

Plugin Interfaces in LIS

6 Plugin interfaces in LIS

The plugin interfaces in LIS are the functional abstractions to represent various user-defined components in LIS. The plugins contain the hot spots or extensible interfaces for incorporating new domains, running modes, LSMS, meteorological analyses, data assimilation tools, and parameters. These adaptable interfaces in LIS enable the reuse of the broad set of data, high performance computing, data management, visualization tools, and the land modeling infrastructure in LIS.

6.1 Fortran: Module Interface runmode_pluginMod (Source File: runmode_pluginMod.F90)

This module contains the definition of the functions used for LIS initialization, execution, and finalization for different running modes in LIS

REVISION HISTORY:

21 Oct 05 Sujay Kumar Initial Specification

6.1.1 runmode_plugin (Source File: runmode_pluginMod.F90)

This is a custom-defined plugin point for introducing a new running mode. The interface mandates that the following routines be implemented and registered for each of the running modes in LIS

Initialization Defining the init routines needed for each running mode in LIS. (to be registered using `registerlisinit` and later called using `lisinit`)

Run Define the execution routines needed for each running mode in LIS (to be registered using `registerlisrun` and later called using `lisrun`)

Finalize Define the finalize routines needed for each LIS running mode. (to be registered using `registerlisfinalize` and later called using `lisfinalize`)

The user-defined functions are included in the registry using a single index. For example, consider the definition of a retrospective running mode in LIS. The methods should be defined in the registry as follows, if the index of the mode is defined to be 1.

```
call registerlisinit(1,lisinit_retrospective)
call registerlisrun(1,lisrun_retrospective)
call registerlisfinalize(1,lisfinal_retrospective)
```

The functions registered above are invoked using generic calls as follows:

```
call lisinit(1) - calls lisinit_retrospective
call lisrun(1) - calls lisrun_retrospective
call lisfinal(1) - calls lisfinal_retrospective
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call lisinit(lis%runmode)
call lisrun(lis%runmode)
call lisfinal(lis%runmode)
```

where `lis%runmode` is set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine runmode_plugin
```

6.2 Fortran: Module Interface domain_pluginMod (Source File: domain_pluginMod.F90)

This module contains the definition of the functions used for defining routines that initialize various LIS-domains. The user defined functions are incorporated into the appropriate registry to be later invoked through generic calls.

REVISION HISTORY:

17 Feb 2004; Sujay Kumar Initial Specification

6.2.1 domain_plugin (Source File: domain_pluginMod.F90)

This is a plugin point for introducing a new LIS-domain. The interface mandates that the following interfaces be implemented and registered for each LIS-domain.

read input Routines to read domain specific options and accordingly set up domain decomposition. (to be registered using **registerinput** and later invoked through **readinput** method)

create grid and tile spaces Routines to create the grid and tile spaces (to be registered using **registerdomain** and later invoked through **makedomain** method)

The user-defined functions are included in the registry using a single index. For example, consider a domain definition using lat/lon projection with a SW to NE data ordering. The methods should be defined in the registry as follows, if the index of the domain is defined to be 1.

```
call registerinput(1,readinput_latlon)
call registerdomain(1,createtiles_latlon)
```

The functions registered above are invoked using generic calls as follows:

```
call readinput(1) - calls readinput_latlon
call makedomain(1) - calls createtiles_latlon
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call readinput(lis%domain)
call readobservations(lis%daobs)
```

where *lis%domain* is set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine domain_plugin
```

6.3 Fortran: Module Interface baseforcing_pluginMod (Source File: baseforcing_pluginMod.F90)

This module contains the definition of the functions used for incorporating a new base forcing scheme. The user defined functions are incorporated into the appropriate registry to be later invoked through generic calls.

REVISION HISTORY:

11 Dec 03 Sujay Kumar Initial Specification

6.3.1 baseforcing_plugin (Source File: baseforcing_pluginMod.F90)

INTERFACE:

```
subroutine baseforcing_plugin
```

DESCRIPTION:

This is a plugin point for introducing a new base forcing scheme. The interface mandates that the following interfaces be implemented and registered for each base forcing scheme.

retrieval of forcing data Routines to retrieve forcing data and to interpolate them. (to be registered using `registerget` and later invoked through `retrieveforcing` method)

definition of native domain Routines to define the native domain (to be registered using `registerdefineNative` and later invoked through `defineNative` method)

temporal interpolation Interpolate forcing data temporally. (to be registered using `registerTimeInterp` and later invoked through `timeinterp` method)

The index used in the register calls should be used to select the appropriate base forcing scheme. For example, assume that the GDAS forcing scheme is incorporated in the registry by the following calls.

```
call registerget(1,getgdas)
call registerdefineNative(1,defineNativeGDAS)
call registerTimeInterp(1,time_interp_gdas)
call registerforcingfinal(1,gdasforcing_finalize)
```

The index used here to register these functions is 1. To invoke these methods, the corresponding calls will be:

```
call retrieveforcing(1) - calls getgdas
call defineNative(1) - calls defineNativeGDAS
call timeinterp(1) - calls time_interp_gdas
call forcingfinalize(1) - calls gdasforcing_finalize
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call retrieveforcing(lis%force,n)
call defineNative(lis%force)
call timeinterp(lis%force,n)
call forcingfinalize(lis%force)
```

where `lis%force` is set through the configuration utility, enabling the user to select any of the base forcing schemes, at runtime.

6.4 Fortran: Module Interface suppforcing_pluginMod (Source File: suppforcing_pluginMod.F90)

This module contains the definition of the functions used for incorporating a new supplemental forcing scheme. The user defined functions are incorporated into the appropriate registry to be later invoked through generic calls.

REVISION HISTORY:

6.4.1 suppforcing_plugin (Source File: suppforcing_pluginMod.F90)

This is a plugin point for introducing a new supplemental forcing scheme. The interface mandates that the following interfaces be implemented and registered for each base forcing scheme.

retrieval of forcing data Routines to retrieve forcing data and to interpolate them. (to be registered using `registerreadsupp` and later invoked through `getsuppforc` method)

definition of native domain Routines to define the native domain (to be registered using `registerdefinenativesupp` and later invoked through `defineNativesupp` method)

temporal interpolation Interpolate forcing data temporally. (to be registered using `registersuppti` and later invoked through `timeinterpsupp` method)

Finalize Cleanup allocated structures (to be registered using `registersuppforcingfinal` and later invoked through `suppforcingfinalize` method)

The index used in the register calls should be used to select the appropriate base forcing scheme. For example, assume that the CMAP precipitation forcing scheme is incorporated in the registry by the following calls.

```
call registerreadsupp(1,getcmap)
call registerdefinenativesupp(1,defineNativeCMAP)
call registersuppti(1,time_interp_cmap)
call registersuppforcingfinal(1,cmapforcing_finalize)
```

The index used here to register these functions is 1. To invoke these methods, the corresponding calls will be:

```
call getsuppforc(1) - calls getcmap
call defineNativesupp(1) - calls defineNativeCMAP
call timeinterpsupp(1) - calls time_interp_cmap
call suppforcingfinalize(1) - calls cmapforcing_finalize
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call getsuppforc(lis%suppforce)
call defineNativeSupp(lis%suppforce)
call timeinterpsupp(lis%suppforce)
call suppforcingfinalize(lis%suppforce)
```

where `lis%suppforce` is set through the configuration utility, enabling the user to select any of the base forcing schemes, at runtime. INTERFACE:

```
subroutine suppforcing_plugin
```

6.5 Fortran: Module Interface lsm_pluginMod (Source File: lsm_pluginMod.F90)

This module contains the definition of the functions used for land surface model initialization, execution, reading and writing of restart files and other relevant land surface model computations, corresponding to each of the LSMs used in LIS.

REVISION HISTORY:

09 Oct 03 Sujay Kumar Initial Specification

6.5.1 lsm_plugin (Source File: lsm_pluginMod.F90)

This is a custom-defined plugin point for introducing a new LSM. The interface mandates that the following routines be implemented and registered for each of the LSM that is included in LIS.

Initialization Definition of LSM variables (to be registered using `registerlsmmini` and later called using `lsmmini`)

Setup Initialization of parameters (to be registered using `registerlsmsetup` and later called using `lsmsetup`)

Run Routines to execute LSM on a single gridcell for single timestep (to be registered using `registerlsmrun` and later called using `lsmrun`)

Read restart Routines to read a restart file for an LSM run (to be registered using `registerlsmrestart` and later called using `lsmrestart`)

Output Routines to write output (to be registered using `registerlsmoutput` and later called using `lsmoutput`)

Forcing transfer to model tiles Routines to transfer an array of given forcing to model tiles (to be registered using `registerlsmf2t` and later called using `lsmf2t`)

Write restart Routines to write a restart file (to be registered using `registerlsmwrst` and later called using `lsmwrst`)

Finalize Routines to cleanup LSM data structures (to be registered using `registerlsmfinalize` and later called using `lsmfinalize`)

There are few other interfaces that needs to be specified if the LSM is used in the following modes.
If used for a coupled run with an atmospheric component:

Specify an export state Routines to specify the variables to be sent to the atmos. component (to be registered using `registerlsmsetexport` and later called using `lsmsetexport`)

If used for data assimilation:

Define the number of prognostic variables Routine that returns the number of prognostic variables to be used (to be registered using `registergetnpr` and later called through `getnpr` method)

Reset Routine that resets the LSM back to a previously saved state (to be registered using `registerlsmreset` and later called through `lsmreset` method)

Return the state variable Routine that retrieves the specified state variable(s) (to be registered using `registergetstatevar` through `getstatevar` method)

Set the state variable Routine that sets the specified state variable(s) (to be registered using `registersetstatevar` through `setstatevar` method)

Transform observations Routine that translates the observations to model variables (to be registered using `registerobstransform` through `obstransform` method)

The user-defined functions are included in the registry using a user-selected indices. For example, consider the Noah LSM is incorporated in the registry by the following calls. In case of registry functions defined with two indices, the first index refers to Noah LSM and the second index refers to the variable being assimilated (in the following example, soil moisture)

```

call registerlsmmini(1,noah_varder_ini)
call registerlsmsetup(1,noah_setup)
call registerlsmf2t(1,noah_f2t)
call registerlsmrun(1,noah_main)
call registerlsmrestart(1,noahrst)
call registerlsmdynsetup(1,noah_dynsetup)
call registerlsmoutput(1,noah_output)
call registerlsmwrst(1,noah_writerst)
call registerlsmfinalize(1,noah_finalize)
call registerlsmexport(1,noah_setexport)
call registergetnpr(1,noah_getnpr)
call registerlsmreset(1,noah_reset)
call registergetstatevar(1,1,noah_getsoilm)
call registersetstatevar(1,1,noah_setsoilm)
call registerobstransform(1,1,noah_obs2st_vsm)

```

The functions registered above are invoked using generic calls as follows:

```

call lsmmini(1)      - calls noah_varder_ini
call lsmsetup(1)     - calls noah_setup
call lsmf2t(1)       - calls noah_f2t
call lsmrun(1)        - calls noah_main
call lsmdynsetup(1)   - calls noah_dynsetup
call lsmoutput(1)     - calls noah_output
call lsmwrst(1)       - calls noah_writerst
call lsmfinalize(1)   - calls noah_finalize
call getnpr(1)        - calls noah_getnpr
call lsmreset(1)      - calls noah_reset
call getstatevar(1,1)  - calls noah_getsoilm
call setstatevar(1,1)  - calls noah_setsoilm
call obstransform(1,1) - calls noah_obs2st_vsm

```

In the LIS code, the above calls are typically invoked in the following manner.

```

call lsmmini(lis%lsm)
call lsmsetup(lis%lsm)
call lsmf2t(lis%lsm)
call lsmrun(lis%lsm)
call lsmdynsetup(lis%lsm)
call lsmoutput(lis%lsm)
call lsmwrst(lis%lsm)
call lsmfinalize(lis%lsm)
call getnpr(lis%lsm)
call lsmreset(lis%lsm)
call getstatevar(lis%lsm, lis%daobs)
call setstatevar(lis%lsm, lis%daobs)
call obstransform(lis%lsm, lis%daobs)

```

where *lis%lsm* and *lis%daobs* are set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine lsm_plugin
```

6.6 Fortran: Module Interface param_pluginMod (Source File: param_pluginMod.F90)

This module contains the definition of the functions used for defining routines to read various sources of parameters maps. The user defined functions are incorporated into the appropriate registry to be later invoked through generic calls.

REVISION HISTORY:

11 Dec 03 Sujay Kumar Initial Specification

6.6.1 topo_plugin (Source File: param_pluginMod.F90)

This is a plugin point for introducing new topography datasets. The interface mandates that the following routines be implemented and registered for each parameter data source.

read the elevation data Routines to retrieve the elevation data (to be registered using `registerreadelev` and later called using the generic `readelev` method)

read the slope data Routines to retrieve the slope data (to be registered using `registerreadslope` and later called using the generic `readslope` method)

read the aspect data Routines to retrieve the aspect data (to be registered using `registerreadaspect` and later called using the generic `readaspect` method)

read the curvature data Routines to retrieve the curvature data (to be registered using `registerreadcurv` and later called using the generic `readcurv` method)

The user-defined functions are included in the registry using two indices. For example, consider the incorporation of the topography datasets from the GTOPO30 source with LIS using a latlon projection. The methods should be defined in the registry as follows, if the index of the source is defined to 1 and latlon projection in LIS uses and index of 1

```
call registerreadelev(1,1,read_elev_gtopo30)
call registerreadslope(1,1,read_slope_gtopo30)
call registerreadaspect(1,1,read_aspect_gtopo30)
call registerreadcurv(1,1,read_curv_gtopo30)
```

The functions registered above are invoked using generic calls as follows:

```
call readelev(1,1) - calls read_elev_gtopo30
call readslope(1,1) - calls read_slope_gtopo30
call readaspect(1,1) - calls read_aspect_gtopo30
call readcurv(1,1) - calls read_curv_gtopo30
```

In the LIS code, the above calls are typically invoked in the following manner.

```

call readelev(lis%domain, lis%toposrc)
call readslope(lis%domain, lis%toposrc)
call readaspect(lis%domain, lis%toposrc)
call readcurv(lis%domain, lis%toposrc)

```

where *lis%domain* and *lis%toposrc* are set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine topo_plugin
```

6.6.2 soils_plugin (Source File: param_pluginMod.F90)

This is a plugin point for introducing new soil parameter datasets. The interface mandates that the following routines be implemented and registered for each parameter data source.

read the texture data Routines to retrieve the texture data (to be registered using `registerreadsoiltexture` and later called using the generic `readtexture` method)

read the sand fraction data Routines to retrieve the sand fraction data (to be registered using `registerreadsand` and later called using the generic `readsand` method)

read the clay fraction data Routines to retrieve the clay fraction data (to be registered using `registerreadclay` and later called using the generic `readclay` method)

read the silt fraction data Routines to retrieve the silt fraction data (to be registered using `registerreadsilt` and later called using the generic `readsilt` method)

read the soil color data Routines to retrieve the soil color data (to be registered using `registerreadcolor` and later called using the generic `readcolor` method)

read the soil porosity data Routines to retrieve the soil porosity data (to be registered using `registerreadporosity` and later called using the generic `readporosity` method)

read the saturated matric potential data Routines to retrieve the saturated matric potential data (to be registered using `registerreadpsisat` and later called using the generic `readpsisat` method)

read the hydraulic conductivity data Routines to retrieve the hydraulic conductivity data (to be registered using `registerreadksat` and later called using the generic `readksat` method)

read the b parameter data Routines to retrieve the b parameter data (to be registered using `registerreadbexp` and later called using the generic `readbexp` method)

read the quartz data Routines to retrieve the quartz data (to be registered using `registerreadquartz` and later called using the generic `readquartz` method)

The user-defined functions are included in the registry using two indices. For example, consider the incorporation of the soil datasets from the FAO source with LIS using a latlon projection. The methods should be defined in the registry as follows (only a subset of the parameters is shown below, others can be defined in the same manner), if the index of the source is defined to 1 and latlon projection in LIS uses and index of 1

```

call registerreadsand(1,1,read_faosand)
call registerreadclay(1,1,read_faoclay)
call registerreadsilt(1,1,read_faosilt)

```

The functions registered above are invoked using generic calls as follows:

```
call readsand(1,1) - calls read_faosand  
call readclay(1,1) - calls read_faoclay  
call readsilt(1,1) - calls read_faosilt
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call readsand(lis%domain, lis%soilsrc)  
call readclay(lis%domain, lis%soilsrc)  
call readsilt(lis%domain, lis%soilsrc)
```

where *lis%domain* and *lis%soilsrc* are set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine soils_plugin
```

6.6.3 **laisai_plugin** (Source File: param_pluginMod.F90)

This is a plugin point for introducing new LAI/SAI datasets. The interface mandates that the following routines be implemented and registered for each parameter data source.

read the LAI data Routines to retrieve the LAI data (to be registered using `registerreadlai` and later called using the generic `readlai` method)

read the SAI data Routines to retrieve the SAI data (to be registered using `registerreadsai` and later called using the generic `readsai` method)

The user-defined functions are included in the registry using two indices. For example, consider the incorporation of the LAI/SAI datasets from the AVHRR source with LIS using a latlon projection. The methods should be defined in the registry as follows if the index of the source is defined to 1 and latlon projection in LIS uses and index of 1

```
call registerreadlai(1,1,read_ll_avhrrlai)  
call registerreadsai(1,1,read_ll_avhrrsai)
```

The functions registered above are invoked using generic calls as follows:

```
call readlai(1,1) - calls read_ll_avhrrlai  
call readsai(1,1) - calls read_ll_avhrrsai
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call readlai(lis%domain,lislaisrc)  
call readsai(lis%domain,lislaisrc)
```

where *lis%domain* and *lis%laisrc* are set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine laisai_plugin
```

6.6.4 landcover_plugin (Source File: param_pluginMod.F90)

This is a plugin point for introducing new landcover datasets. The interface mandates that the following routines be implemented and registered for each parameter data source.

read the landmask data Routines to retrieve the landmask data (to be registered using `registerreadmask` and later called using the generic `readmask` method)

read the landcover data Routines to retrieve the landcover data (to be registered using `registerreadlc` and later called using the generic `readlandcover` method)

The user-defined functions are included in the registry using two indices. For example, consider the incorporation of the landcover datasets from the UMD source with LIS using a latlon projection. The methods should be defined in the registry as follows if the index of the source is defined to 1 and latlon projection in LIS uses and index of 1

```
call registerreadmask(1,1,read_latlon_umdmask)
call registerreadlc(1,1,read_latlon_umdlc)
```

The functions registered above are invoked using generic calls as follows:

```
call readmask(1,1) - calls read_latlon_umdmask
call readlc(1,1) - calls read_latlon_umdlc
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call readmask(lis%domain,lis%vegsrc)
call readlandcover(lis%domain,lis%vegsrc)
```

where `lis%domain` and `lis%vegsrc` are set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine landcover_plugin
```

6.6.5 tbot_plugin (Source File: param_pluginMod.F90)

This is a plugin point for introducing new bottom temperature datasets. The interface mandates that the following routines be implemented and registered for each parameter data source.

read the bottom temperature data Routines to retrieve the bottom temperature data (to be registered using `registerreadtbot` and later called using the generic `readtbot` method)

The user-defined functions are included in the registry using a single index. For example, consider the incorporation of the tbot datasets to be used in LIS running lat/lon projection. The methods should be defined in the registry as follows, if the index of the domain is defined to 1.

```
call registerreadtbot(1,read_statictbot)
```

The functions registered above are invoked using generic calls as follows:

```
call readtbot(1) - calls read_statictbot
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call readtbot(lis%domain)
```

where *lis%domain* is set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine tbot_plugin
```

6.6.6 gfrac_plugin (Source File: param_pluginMod.F90)

This is a plugin point for introducing new greenness datasets. The interface mandates that the following routines be implemented and registered for each parameter data source.

read the greenness data Routines to retrieve the greenness data (to be registered using `registerreadgfrac` and later called using the generic `readgfrac` method)

The user-defined functions are included in the registry using two indices. For example, consider the incorporation of the greenness datasets from the NCEP source with LIS using a latlon projection. The methods should be defined in the registry as follows if the index of the source is defined to 1 and latlon projection in LIS uses and index of 1

```
call registerreadgfrac(1,1,read_llgfrac)
```

The functions registered above are invoked using generic calls as follows:

```
call readgfrac(1,1) - calls read_llgfrac
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call readgfrac(lis%domain,lis%gfracsrc)
```

where *lis%domain* and *lis%gfracsrc* are set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine gfrac_plugin
```

6.6.7 alb_plugin (Source File: param_pluginMod.F90)

This is a plugin point for introducing new albedo datasets. The interface mandates that the following routines be implemented and registered for each parameter data source.

read the max snow albedo data Routines to retrieve the max snow albedo data (to be registered using `registerreadmxsnoalb` and later called using the generic `readmxsnoalb` method)

read the albedo climatology data Routines to retrieve the albedo climatology data (to be registered using `registerreadalbedo` and later called using the generic `readalbedo` method)

The user-defined functions are included in the registry using two indices. For example, consider the incorporation of the albedo datasets from the NCEP source with LIS using a latlon projection. The methods should be defined in the registry as follows if the index of the source is defined to 1 and latlon projection in LIS uses and index of 1

```
call registerreadmxsnoalb(1,1,read_llmxsnoalb)
call registerreadalbedo(1,1,read_llalbedo)
```

The functions registered above are invoked using generic calls as follows:

```
call readmxsnoalb(1,1) - calls read_llmxsnoalb
call readalbedo(1,1)    - calls read_llalbedo
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call readmxsnoalb(lis%domain,lis%albedosrc)
call readalbedo(lis%domain,lis%albedoarc)
```

where *lis%domain* and *lis%albedosrc* are set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine alb_plugin
```

6.6.8 snow_plugin (Source File: param_pluginMod.F90)

This is a plugin point for introducing new snow datasets. The interface mandates that the following routines be implemented and registered for each parameter data source.

read the snowdepth data Routines to retrieve the snow depth data (to be registered using `registerreadsnowdepth` and later called using the generic `readsnowdepth` method)

The user-defined functions are included in the registry using two indices. For example, consider the incorporation of the snowdepth datasets from AFWA with LIS using a latlon projection. The methods should be defined in the registry as follows if the index of the source is defined to 1 and latlon projection in LIS uses and index of 1

```
call registerreadsnowdepth(1,1,read_llsnowdepth)
```

The functions registered above are invoked using generic calls as follows:

```
call readsnowdepth(1,1) - calls read_llsnowdepth
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call readsnowdepth(lis%domain,lis%snowsrc)
```

where *lis%domain* and *lis%snowsrc* are set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine snow_plugin
```

6.7 Fortran: Module Interface `dataassim_pluginMod.F90` (Source File: `dataassim_pluginMod.F90`)

This module contains the definition of the functions used for defining routines that performs data assimilation. The user defined functions are incorporated into the appropriate registry to be later invoked through generic calls.

REVISION HISTORY:

27 Feb 2005; Sujay Kumar Initial Specification

6.7.1 `dataassim_plugin` (Source File: `dataassim_pluginMod.F90`)

This is a plugin point for introducing a new data assimilation scheme. As explained in the `dataassim_module`, there are many different levels of abstractions associated with the data assimilation implementation. The implementations defined in this subroutine complete the "wirings" required to complete the data assimilation algorithm-related abstractions. Other required interfaces defined in `lsm_pluginMod` and `dataobs_pluginMod` should be completed to complete a successful implementation.

The following interfaces should be implemented in this routine.

Setup Initialization of data and memory structures (to be registered using `registerdasetup` and later called through `dasetup`)

Forecast Routines to compute model forecast error covariances (to be registered using `registerfrcst` and later called through `forecast`)

Assimilate Routines to perform data assimilation using observations. (to be registered using `registerdaassim` and later called through `assimilate`)

There are two indices used to register a routine. The first index corresponds to the type of data assimilation algorithm used, and the second index represents the variable being assimilated. For example, consider an instance where soil moisture assimilation using Ensemble Kalman Filter (EnKF) is conducted and assume that EnKF is denoted by an index 3 and soil moisture is denoted by an index 1. The methods should be defined in the registry as follows.

```
call registerdasetup(3,1,sm_enkf_init)
call registerfrcst(3,1,sm_enkf_frcst)
call registerassim(3,1,sm_enkf_assim)
```

The functions registered above are invoked using generic calls as follows:

```
call dasetup(3,1) - calls sm_enkf_init
call frcst(3,1)   - calls sm_enkf_frcst
call assimilate(3,1) - calls sm_enkf_assim
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call dasetup(lis%daalg,lis%davar)
call frcst(lis%daalg,lis%davar)
call assimilate(lis%daalg,lis%davar)
```

where `lis%daalg` and `lis%davar` are set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine dataassim_plugin
```

6.8 Fortran: Module Interface dataobs_pluginMod (Source File: dataobs_pluginMod.F90)

This module contains the definition of the functions that are used to read observation data for data assimilation. The user defined functions are incorporated into the appropriate registry to be later invoked through generic calls.

REVISION HISTORY:

27 Feb 2005; Sujay Kumar Initial Specification

6.8.1 dataobs_plugin (Source File: dataobs_pluginMod.F90)

This is a plugin point for introducing a routines to handle the observation data for assimilation. As explained in the `dataassim_module`, there are many different levels of abstractions associated with the data assimilation implementation. The implementations defined in this subroutine complete the "wirings" required to complete the observation data-related abstractions. Other required interfaces defined in `lsm_pluginMod` and `dataassim_pluginMod` should be completed to complete a successful implementation.

The following interfaces should be implemented in this routine.

Setup Initialization of data and memory structures (to be registered using `registerdaobssetup` and later called through `daobssetup`)

readobservations Routines to read the observation data and perform any spatial transformation. (to be registered using `registerdaobs` and later called through `readobservations`)

The user-defined functions are included in the registry using a single index. For example, consider an instance where soil moisture assimilation using TMI soil moisture data is conducted. The methods should be defined in the registry as follows, if the index of the TMI data is defined to be 1.

```
call registerdaobssetup(1,smobs_setup)
call registerdaobs(1,read_tmism)
```

The functions registered above are invoked using generic calls as follows:

```
call daobssetup(1)      -  calls smobs_setup
call readobservations(1) -  calls read_tmism
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call daobssetup(lis%daobs)
call readobservations(lis%daobs)
```

where `lis%daobs` is set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine dataobs_plugin
```

6.9 Fortran: Module Interface perturb_pluginMod (Source File: perturb_pluginMod.F90)

This module contains the definition of the functions used for defining routines that perform perturbations of model states, forcing, or parameters. The user defined functions are incorporated into the appropriate registry to be later invoked through generic calls.

REVISION HISTORY:

08Jul2005; Sujay Kumar Initial Specification

INTERFACE:

6.9.1 perturb_plugin (Source File: perturb_pluginMod.F90)

This is a plugin point for introducing a new forcing perturbation scheme. The interface mandates that the following interfaces be implemented for each scheme.

Setup Initialization of data and memory structures (to be registered using `registerperturbsetup` and later called using the generic call `perturbinit`)

Forecast Routines to compute perturbations (to be registered using `registerperturb` and later called using the generic call `perturb`)

The user-defined functions are included in the registry using a single index. For example, consider a new scheme called 'FOO' The methods should be defined in the registry as follows, if the index of the scheme is defined to be 1.

```
call registerperturbsetup(1,foo_setup)
call registerperturb(1,foo_method)
```

The functions registered above are invoked using generic calls as follows:

```
call perturbinit(1) - calls foo_setup
call perturb(1)      - calls foo_method
```

In the LIS code, the above calls are typically invoked in the following manner.

```
call perturbinit(lis%perturb)
call perturb(lis%perturb)
```

where `lis%pertrub` is set through the configuration utility, enabling the user make a selection at runtime.

INTERFACE:

```
subroutine perturb_plugin
```

Part V

Interpolation Tools in LIS

7 Interpolation tools in LIS

The spatial interpolation tools in LIS are used for the geospatial transformation of an input data to the LIS grid. The tools in LIS are based on the ipolates package developed at NCEP. Currently LIS supports the transformation and interpolation of polar stereographic, lambert conformal, and mercator grid projection data on to an equidistant cylindrical (lat/lon) grid projection. Bilinear, conservative and neighbor search interpolations are supported.

7.0.2 bilinear_interp_input (Source File: bilinear_interp_input.F90)

INTERFACE:

```
subroutine bilinear_interp_input (gridDesci,gridDesco,npts,&
    rlat,rlon,n11,n12,n21,n22,w11,w12,w21,w22)

    implicit none
```

ARGUMENTS:

```
real, intent(in)      :: gridDesci(50)
real                  :: gridDesco(50)
integer               :: npts
real                  :: rlat(npts)
real                  :: rlon(npts)
integer               :: n11(npts),n12(npts),n21(npts),n22(npts)
real                  :: w11(npts),w12(npts),w21(npts),w22(npts)
```

DESCRIPTION:

This subprogram performs issues calls to compute the interpolation weights and neighbor information for bilinear interpolation, from any grid to any grid for scalar fields. The grids are defined by their grid description arrays.

The grid description arrays are based on the decoding schemes used by NCEP. However, in order to remove the integer arithmetic employed in the original ipolates, the routines are rewritten using real number manipulations. The general structure remains the same.

The current code recognizes the following projections:

(gridDesc(1)=0) equidistant cylindrical
(gridDesc(1)=1) mercator cylindrical
(gridDesc(1)=3) lambert conformal conical
(gridDesc(1)=4) gaussian cylindrical (spectral native)
(gridDesc(1)=5) polar stereographic azimuthal

where gridDesc could be defined for either the input grid or the output grid.

The arguments are:

gridDesci input grid description parameters

gridDesco output grid description parameters

npts number of points to in the output field

rlat output latitudes in degrees

rlon output longitudes in degrees

w11,w12,w21,w22 weights to be used for interpolation

n11,n12,n21,n22 index of neighbor points

The routines invoked are:

compute_earth_coord (7.0.8)

Computes the earth coordinates for the output grid

compute_grid_coord (7.0.14)

Computes the grid coordinates of the input grid, based on the earth coordinates of the output grid.

get_field_pos (7.0.20)

computes the field position for a given point

7.0.3 bilinear_interp (Source File: bilinear_interp.F90)

REVISION HISTORY:

04-10-96 Mark Iredell; Initial Specification

05-27-04 Sujay Kumar : Modified verision with floating point arithmetic

INTERFACE:

```
subroutine bilinear_interp(gridDesco,ibi,li,gi,ibo,lo,go,mi,mo, &
    rlat,rlon,w11,w12,w21,w22,n11,n12,n21,n22,udef,iret)
```

USES:

```
implicit none
```

ARGUMENTS:

```
real      :: gridDesco(50)
integer   :: ibi
integer   :: ibo
integer   :: mi
integer   :: mo
logical*1 :: li(mi)
logical*1 :: lo(mo)
real      :: gi(mi)
real      :: go(mo)
real      :: rlat(mo)
real      :: rlon(mo)
real      :: w11(mo),w12(mo), w21(mo),w22(mo)
integer   :: n11(mo),n12(mo),n21(mo),n22(mo)
real      :: udef
integer   :: iret
```

DESCRIPTION:

This subprogram performs bilinear interpolation from any grid to any grid for scalar fields. The routine is based on the spatial interpolation package ipolates from NCEP.

The algorithm simply computes (weighted) averages of bilinearly interpolated points arranged in a square box centered around each output grid point and stretching nearly halfway to each of the neighboring grid points. the grids are defined by their grid description arrays.

The grid description arrays are based on the decoding schemes used by NCEP. However, in order to remove the integer arithmetic employed in the original ipolates, the routines are rewritten using real number manipulations. The general structure remains the same.

The current code recognizes the following projections:

(gridDesc(1)=0) equidistant cylindrical
 (gridDesc(1)=1) mercator cylindrical

(gridDesc(1)=3) lambert conformal conical
(gridDesc(1)=4) gaussian cylindrical (spectral native)
(gridDesc(1)=5) polar stereographic azimuthal

where gridDesc could be defined for either the input grid or the output grid. The routine also returns the the number of output grid points and their latitudes and longitudes are also returned. The input bitmaps will be interpolated to output bitmaps. output bitmaps will also be created when the output grid extends outside of the domain of the input grid. the output field is set to 0 where the output bitmap is off.
The arguments are:

gridDesco output grid description parameters

ibi integer input bitmap flags

li logical input bitmaps

gi real input fields to interpolate

ibo integer output bitmap flags

lo logical output bitmaps

go real output fields interpolated

mi integer dimension of input grid fields

mo integer dimension of output grid fields

rlat output latitudes in degrees

rlon output longitudes in degrees

w11,w12,w21,w22 weights to be used for interpolation

n11,n12,n21,n22 index of neighbor points

udef undefined value to be used

iret return code (0-success)

The routines invoked are:

polfixs (7.0.21)

 Apply corrections for poles

7.0.4 conserv_interp_input (Source File: conserv_interp_input.F90)

INTERFACE:

```
subroutine conserv_interp_input(gridDesci,gridDesco,npts,&
    rlat2,rlon2,n112,n122,n212,n222,w112,w122,w212,w222)
```

```
    implicit none
```

ARGUMENTS:

```

real, intent(in)    :: gridDesci(50)
real                :: gridDesco(50)
integer             :: npts
real                :: rlat2(npts)
real                :: rlon2(npts)
integer             :: n112(npts,25),n122(npts,25),&
                     n212(npts,25),n222(npts,25)
real                :: w112(npts,25),w122(npts,25),&
                     w212(npts,25),w222(npts,25)

```

DESCRIPTION:

This subprogram performs issues calls to compute the interpolation weights and neighbor information for budget bilinear interpolation, from any grid to any grid for scalar fields. The grids are defined by their grid description arrays.

The grid description arrays are based on the decoding schemes used by NCEP. However, in order to remove the integer arithmetic employed in the original ipolates, the routines are rewritten using real number manipulations. The general structure remains the same.

The current code recognizes the following projections:

- (gridDesc(1)=0) equidistant cylindrical
- (gridDesc(1)=1) mercator cylindrical
- (gridDesc(1)=3) lambert conformal conical
- (gridDesc(1)=4) gaussian cylindrical (spectral native)
- (gridDesc(1)=5) polar stereographic azimuthal

where gridDesc could be defined for either the input grid or the output grid.

The arguments are:

gridDesci input grid description parameters

gridDesco output grid description parameters

npts number of points to in the output field

rlat2 output latitudes in degrees

rlon2 output longitudes in degrees

w112,w122,w212,w222 weights to be used for interpolation

n112,n122,n212,n222 index of neighbor points

The routines invoked are:

compute_earth_coord (7.0.8)

Computes the earth coordinates for the output grid

compute_grid_coord (7.0.14)

Computes the grid coordinates of the input grid, based on the earth coordinates of the output grid.

get_field_pos (7.0.20)

computes the field position for a given point

7.0.5 conserv_interp (Source File: conserv_interp.F90)

REVISION HISTORY:

04-10-96 Mark Iredell; Initial Specification
05-27-04 Sujay Kumar : Modified version with floating point arithmetic

INTERFACE:

```
subroutine conserv_interp(gridDesc, ibi, li, gi, ibo, lo, go, mi, mo, &
    rlat, rlon, w11, w12, w21, w22, n11, n12, n21, n22, udef, iret)
    implicit none
```

ARGUMENTS:

```
integer, parameter :: nb3 = 25, nb4 = 25
real      :: gridDesc(50)
integer   :: ibi
integer   :: ibo
integer   :: mi
integer   :: mo
logical*1 :: li(mi)
logical*1 :: lo(mo)
real      :: gi(mi)
real      :: go(mo)
real      :: rlat(mo)
real      :: rlon(mo)
integer   :: n11(mo,nb4),n21(mo,nb4),n12(mo,nb4),n22(mo,nb4)
real      :: w11(mo,nb4),w21(mo,nb4),w12(mo,nb4),w22(mo,nb4)
real      :: udef
integer   :: iret
```

DESCRIPTION:

This subprogram performs budget interpolation from any grid to any grid for scalar fields. The routine is based on the spatial interpolation package ipolates from NCEP.

The algorithm simply computes (weighted) averages of bilinearly interpolated points arranged in a square box centered around each output grid point and stretching nearly halfway to each of the neighboring grid points, using 25 points for computing the average.

The grid description arrays are based on the decoding schemes used by NCEP. However, in order to remove the integer arithmetic employed in the original ipolates, the routines are rewritten using real number manipulations. The general structure remains the same.

The current code recognizes the following projections:

(gridDesc(1)=0) equidistant cylindrical
(gridDesc(1)=1) mercator cylindrical
(gridDesc(1)=3) lambert conformal conical
(gridDesc(1)=4) gaussian cylindrical (spectral native)
(gridDesc(1)=5) polar stereographic azimuthal

where gridDesc could be defined for either the input grid or the output grid. The routine also returns the the number of output grid points and their latitudes and longitudes are also returned. The input bitmaps will be interpolated to output bitmaps. output bitmaps will also be created when the output grid extends outside of the domain of the input grid. the output field is set to 0 where the output bitmap is off.

The arguments are:

gridDesc output grid description parameters

ibi integer input bitmap flags

li logical input bitmaps

gi real input fields to interpolate

ibo integer output bitmap flags

lo logical output bitmaps
go real output fields interpolated
mi integer dimension of input grid fields
mo integer dimension of output grid fields
rlat output latitudes in degrees
rlon output longitudes in degrees
w11,w12,w21,w22 weights to be used for interpolation
n11,n12,n21,n22 index of neighbor points
undef undefined value to be used
iret return code (0-success)

The routines invoked are:

polfixs (7.0.21)
Apply corrections for poles

7.0.6 neighbor_interp_input (Source File: neighbor_interp_input.F90)

INTERFACE:

```
subroutine neighbor_interp_input (gridDesci,gridDesco,npts, rlat2,rlon2,n112)
  implicit none
```

ARGUMENTS:

```
real, intent(in) :: gridDesci(50)
real :: gridDesco(200)
integer :: npts
real :: rlat2(npts)
real :: rlon2(npts)
integer :: n112(npts)
```

DESCRIPTION:

This subprogram performs issues calls to compute the neighbor information for neighbor search interpolation, from any grid to any grid for scalar fields. The grids are defined by their grid description arrays. The grid description arrays are based on the decoding schemes used by NCEP. However, in order to remove the integer arithmetic employed in the original ipolates, the routines are rewritten using real number manipulations. The general structure remains the same.

The current code recognizes the following projections:

- (gridDesc(1)=0) equidistant cylindrical
- (gridDesc(1)=1) mercator cylindrical
- (gridDesc(1)=3) lambert conformal conical
- (gridDesc(1)=4) gaussian cylindrical (spectral native)
- (gridDesc(1)=5) polar stereographic azimuthal

where gridDesc could be defined for either the input grid or the output grid.

The arguments are:

gridDesci input grid description parameters

gridDesco output grid description parameters

npts number of points to in the output field

rlat2 output latitudes in degrees

rlon2 output longitudes in degrees

n112 index of neighbor points

The routines invoked are:

compute_earth_coord (7.0.8)

Computes the earth coordinates for the output grid

compute_grid_coord (7.0.14)

Computes the grid coordinates of the input grid, based on the earth coordinates of the output grid.

get_field_pos (7.0.20)

computes the field position for a given point

7.0.7 neighbor_interp (Source File: neighbor_interp.F90)

REVISION HISTORY:

04-10-96 Mark Iredell; Initial Specification

05-27-04 Sujay Kumar : Modified version with floating point arithmetic,

INTERFACE:

```
subroutine neighbor_interp(gridDesco,ibi,li,gi,ibo,lo,go,mi,mo, &
    rlat,rlon,n11,udef, iret)
    implicit none
```

ARGUMENTS:

```
real      :: gridDesco(200)
integer   :: ibi
integer   :: ibo
integer   :: mi
integer   :: mo
logical*1 :: li(mi)
logical*1 :: lo(mo)
real      :: gi(mi)
real      :: go(mo)
real      :: rlat(mo)
real      :: rlon(mo)
integer   :: n11(mo)
real      :: udef
integer   :: iret
```

DESCRIPTION:

This subprogram performs neighbor search interpolation from any grid to any grid for scalar fields. The routine is based on the spatial interpolation package ipolates from NCEP.

The algorithm simply performs a nearest neighbor search around each output grid point for interpolation. The grids are defined by their grid description arrays.

The grid description arrays are based on the decoding schemes used by NCEP. However, in order to remove the integer arithmetic employed in the original ipolates, the routines are rewritten using real number manipulations. The general structure remains the same.

The current code recognizes the following projections:

(gridDesc(1)=0) equidistant cylindrical
(gridDesc(1)=1) mercator cylindrical
(gridDesc(1)=3) lambert conformal conical
(gridDesc(1)=4) gaussian cylindrical (spectral native)
(gridDesc(1)=5) polar stereographic azimuthal

where gridDesc could be defined for either the input grid or the output grid. The routine also returns the the number of output grid points and their latitudes and longitudes are also returned. The input bitmaps will be interpolated to output bitmaps. output bitmaps will also be created when the output grid extends outside of the domain of the input grid. the output field is set to 0 where the output bitmap is off.

The arguments are:

gridDesco output grid description parameters

ibi integer input bitmap flags

li logical input bitmaps

gi real input fields to interpolate

ibo integer output bitmap flags

lo logical output bitmaps

go real output fields interpolated

mi integer dimension of input grid fields

mo integer dimension of output grid fields

rlat output latitudes in degrees

rlon output longitudes in degrees

n11 index of neighbor points

udef undefined value to be used

iret return code (0-success)

The routines invoked are:

polfixs (7.0.21)

Apply corrections for poles

7.0.8 compute_earth_coord (Source File: compute_earth_coord.F90)

REVISION HISTORY:

04-10-96 Mark Iredell; Initial Specification

05-27-04 Sujay Kumar; Modified version with floating point arithmetic.

INTERFACE:

```
subroutine compute_earth_coord(gridDesc,npts,fill,xpts,ypts,rlon,rlat,nret)
implicit none
```

ARGUMENTS:

```
real      :: gridDesc(50)
integer   :: npts
real      :: fill
real      :: xpts(npts),ypts(npts)
real      :: rlat(npts)
real      :: rlon(npts)
integer   :: nret
```

DESCRIPTION:

This subroutine computes the earth coordinates (lat/lon values) of the specified domain. This routine is based on the grid decoding routines in the ipolates interoplation package.

The input options include : The current code recognizes the following projections:

(gridDesc(1)=000) equidistant cylindrical
(gridDesc(1)=001) mercator cylindrical
(gridDesc(1)=003) lambert conformal conical
(gridDesc(1)=004) gaussian cylindrical
(gridDesc(1)=005) polar stereographic azimuthal

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts input grid x point coordinates

ypts input grid y point coordinates

rlat output latitudes in degrees

rlon output longitudes in degrees

nret return code (0-success)

The routines invoked are:

compute_earth_coord_latlon (7.0.9)
computes the earth coordinates of a latlon grid

compute_earth_coord_merc (7.0.10)
computes the earth coordinates of a mercator grid

compute_earth_coord_lambert (7.0.11)
computes the earth coordinates of a lambert conformal grid

compute_earth_coord_gauss (7.0.12)
computes the earth coordinates of a gaussian cylindrical grid

compute_earth_coord_polar (7.0.13)
computes the earth coordinates of a polar stereographic grid

7.0.9 compute_earth_coord_latlon (Source File: compute_earth_coord_latlon.F90)

REVISION HISTORY:

04-10-96 Mark Iredell; Initial Specification
05-27-04 Sujay Kumar; Modified version with floating point arithmetic.

INTERFACE:

```
subroutine compute_earth_coord_latlon(gridDesc,npts,fill,xpts,ypts,&
rlon,rlat,nret)

implicit none
```

ARGUMENTS:

real	:: gridDesc(50)
integer	:: npts
real	:: fill
real	:: xpts(npts),ypts(npts)
real	:: rlat(npts)
real	:: rlon(npts)

DESCRIPTION:

This subroutine computes the earth coordinates of the specified domain for an equidistant cylindrical projection. This routine is based on the grid decoding routines in the NCEP interpolation package.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts grid x point coordinates

ypts grid y point coordinates

rlat output latitudes in degrees

rlon output longitudes in degrees

7.0.10 compute_earth_coord_merc (Source File: compute_earth_coord_merc.F90)

REVISION HISTORY:

04-10-96 Mark Iredell; Initial Specification
07-15-05 Sujay Kumar; Modified version with floating point arithmetic.

INTERFACE:

```
subroutine compute_earth_coord_merc(gridDesc,npts,fill,xpts,ypts,&
rlon,rlat,nret)
```

USES:

```
use map_utils
implicit none
```

ARGUMENTS:

```
real          :: gridDesc(50)
integer        :: npts
real          :: fill
real          :: xpts(npts),ypts(npts)
real          :: rlat(npts)
real          :: rlon(npts)
integer        :: nret
```

DESCRIPTION:

This subroutine computes the earth coordinates of the specified domain for a mercator projection. This routine is based on the decoding routines in the NCEP interoplation package and has been modified the adopted module from the Weather Research and Forecasting (WRF) model.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts grid x point coordinates

ypts grid y point coordinates

rlat output latitudes in degrees

rlon output longitudes in degrees

nret return code (0-success)

The routines invoked are:

map_set (7.1.7)

Sets the projection to mercator

ij_to_latlon (7.1.9)

Computes the lat lon values for each i,j

7.0.11 compute_earth_coord_lambert (Source File: compute_earth_coord_lambert.F90)

INTERFACE:

```
subroutine compute_earth_coord_lambert(gridDesc,npts,fill,xpts,ypts,&
                                         rlon,rlat,nret)
```

USES:

```
use map_utils
```

```
implicit none
```

ARGUMENTS:

```
real          :: gridDesc(50)
integer        :: npts
real          :: fill
real          :: xpts(npts),ypts(npts)
real          :: rlat(npts)
real          :: rlon(npts)
integer        :: nret
```

DESCRIPTION:

This subroutine computes the earth coordinates of the specified domain for a lambert conformal projection
This routine is based on the decoding routines in the NCEP interoplation package and has been modified
the adopted module from the Weather Research and Forecasting (WRF) model.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts grid x point coordinates

ypts grid y point coordinates

rlat output latitudes in degrees

rlon output longitudes in degrees

nret return code (0-success)

The routines invoked are:

map_set (7.1.7)

Sets the projection to lambert conformal

ij_to_latlon (7.1.9)

Computes the lat lon values for each i,j

7.0.12 compute_earth_coord_gauss (Source File: compute_earth_coord_gauss.F90)

REVISION HISTORY:

04-10-96 Mark Iredell; Initial Specification

05-27-04 Sujay Kumar; Modified verision with floating point arithmetic.

INTERFACE:

```
subroutine compute_earth_coord_gauss(gridDesc,npts,fill,xpts,ypts,&
rlon,rlat,nret)
```

```
implicit none
```

ARGUMENTS:

```

real          :: gridDesc(50)
integer        :: npts
real          :: fill
real          :: xpts(npts),ypts(npts)
real          :: rlat(npts)
real          :: rlon(npts)

```

DESCRIPTION:

This subroutine computes the earth coordinates of the specified domain for a gaussian cylindrical projection. This routine is based on the grid decoding routines in the NCEP interoplation package.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts grid x point coordinates

ypts grid y point coordinates

rlat output latitudes in degrees

rlon output longitudes in degrees

The routines invoked are:

gausslat (7.0.19)

Computes latitude values in gaussian

7.0.13 compute_earth_coord_polar (Source File: compute_earth_coord_polar.F90)

REVISION HISTORY:

04-10-96 Mark Iredell; Initial Specification

07-15-05 Sujay Kumar; Modified verision with floating point arithmetic.

INTERFACE:

```

subroutine compute_earth_coord_polar(gridDesc,npts,fill,xpts,ypts,&
                                      rlon,rlat,nret)

```

USES:

```

use map_utils
implicit none

```

ARGUMENTS:

```

real          :: gridDesc(50)
integer        :: npts
real          :: fill
real          :: xpts(npts),ypts(npts)
real          :: rlat(npts)
real          :: rlon(npts)
integer        :: nret

```

DESCRIPTION:

This subroutine computes the earth coordinates of the specified domain for a polar stereographic projection. This routine is based on the decoding routines in the NCEP interpolation package and has been modified the adopted module from the Weather Research and Forecasting (WRF) model.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts grid x point coordinates

ypts grid y point coordinates

rlat output latitudes in degrees

rlon output longitudes in degrees

nret return code (0-success)

The routines invoked are:

map_set (7.1.7)

Sets the projection to polar stereographic

ij_to_latlon (7.1.9)

Computes the lat lon values for each i,j

7.0.14 compute_grid_coord (Source File: compute_grid_coord.F90)

REVISION HISTORY:

04-10-96 Mark Iredell; Initial Specification

05-27-04 Sujay Kumar; Modified version with floating point arithmetic.

INTERFACE:

```
subroutine compute_grid_coord(gridDesc,npts,fill,xpts,ypts,rlon,rlat,nret)
  implicit none
```

ARGUMENTS:

```
  real      :: gridDesc(50)
  integer   :: npts
  real      :: fill
  real      :: xpts(npts),ypts(npts)
  real      :: rlat(npts)
  real      :: rlon(npts)
  integer   :: nret
```

DESCRIPTION:

This subroutine computes the grid coordinates (cartesian) of the specified domain. This routine is based on the grid decoding routines in the ipolates interoplation package.

The input options include : The current code recognizes the following projections: (gridDesc(1)=000) equidistant cylindrical (gridDesc(1)=001) mercator cylindrical (gridDesc(1)=003) lambert conformal conical (gridDesc(1)=004) gaussian cylindrical (gridDesc(1)=005) polar stereographic azimuthal

gridDesc grid description parameters
npts integer maximum number of coordinates
fill fill value to set invalid output data
xpts output grid x point coordinates
ypts output grid y point coordinates
rlat input latitudes in degrees
rlon input longitudes in degrees
nret return code (0-success)

The routines invoked are:

compute_grid_coord_latlon (7.0.15)
computes the grid coordinates of a latlon grid
compute_grid_coord_merc (7.0.16)
computes the grid coordinates of a mercator grid
compute_grid_coord_lambert (7.0.17)
computes the grid coordinates of a lambert conformal grid
compute_grid_coord_gauss (7.0.18)
computes the grid coordinates of a gaussian cylindrical grid
compute_grid_coord_polar (7.0.19)
computes the grid coordinates of a polar stereographic grid

7.0.15 compute_grid_coord_latlon (Source File: **compute_grid_coord_latlon.F90**)

REVISION HISTORY:

04-10-96 Mark Iredell; Initial Specification
05-27-04 Sujay Kumar; Modified verision with floating point arithmetic.

INTERFACE:

```
subroutine compute_grid_coord_latlon(gridDesc,npts,fill,xpts,ypts,&
                                      rlon,rlat,nret)

      implicit none
```

ARGUMENTS:

```
real          :: gridDesc(50)
integer        :: npts
real          :: fill
real          :: xpts(npts),ypts(npts)
real          :: rlat(npts)
real          :: rlon(npts)
integer        :: nret
```

DESCRIPTION:

This subroutine computes the grid coordinates of the specified domain for an equidistant cylindrical projection. This routine is based on the grid decoding routines in the NCEP interoplation package.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts output grid x point coordinates

ypts output grid y point coordinates

rlat input latitudes in degrees

rlon input longitudes in degrees

7.0.16 compute_grid_coord_merc (Source File: **compute_grid_coord_merc.F90**)

REVISION HISTORY:

04-10-96 Mark Iredell; Initial Specification

07-15-05 Sujay Kumar; Modified verision with floating point arithmetic.

INTERFACE:

```
subroutine compute_grid_coord_merc(gridDesc,npts,fill,xpts,ypts,&
                                    rlon,rlat,nret)
```

USES:

```
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

real	:: gridDesc(50)
integer	:: npts
real	:: fill
real	:: xpts(npts),ypts(npts)
real	:: rlat(npts)
real	:: rlon(npts)
integer	:: nret

DESCRIPTION:

This subroutine computes the grid coordinates of the specified domain for a mercator projection. This routine is based on the grid decoding routines in the NCEP interoplation package.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data
xpts output grid x point coordinates
ypts output grid y point coordinates
rlat input latitudes in degrees
rlon input longitudes in degrees
!NOTE: This routine is currently unsupported.

7.0.17 compute_grid_coord_lambert (Source File: compute_grid_coord_lambert.F90)

REVISION HISTORY:

04-10-96 Mark Iredell; Initial Specification
07-15-05 Sujay Kumar; Modified version with floating point arithmetic.

INTERFACE:

```
subroutine compute_grid_coord_lambert(gridDesc,npts,fill,xpts,ypts,&
    rlon,rlat,nret)
```

USES:

```
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

```
real          :: gridDesc(50)
integer        :: npts
real          :: fill
real          :: xpts(npts),ypts(npts)
real          :: rlat(npts)
real          :: rlon(npts)
integer        :: nret
```

DESCRIPTION:

This subroutine computes the grid coordinates of the specified domain for a lambert conformal projection. This routine is based on the grid decoding routines in the NCEP interpolation package.

gridDesc grid description parameters
npts integer maximum number of coordinates
fill fill value to set invalid output data
xpts output grid x point coordinates
ypts output grid y point coordinates
rlat input latitudes in degrees
rlon input longitudes in degrees
!NOTE: This routine is currently unsupported.

7.0.18 compute_grid_coord_gauss (Source File: compute_grid_coord_gauss.F90)

REVISION HISTORY:

04-10-96 Mark Iredell; Initial Specification
05-27-04 Sujay Kumar; Modified version with floating point arithmetic.

INTERFACE:

```
subroutine compute_grid_coord_gauss(gridDesc,npts,fill,xpts,ypts,&
                                     rlon,rlat,nret)

      implicit none
```

ARGUMENTS:

```
real           :: gridDesc(50)
integer        :: npts
real           :: fill
real           :: xpts(npts),ypts(npts)
real           :: rlat(npts)
real           :: rlon(npts)
integer        :: nret
```

DESCRIPTION:

This subroutine computes the grid coordinates of the specified domain for a gaussian cylindrical projection. This routine is based on the grid decoding routines in the NCEP interpolation package.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts output grid x point coordinates

ypts output grid y point coordinates

rlat input latitudes in degrees

rlon input longitudes in degrees

The routines invoked are:

gausslat (7.0.19)

Computes latitude values in gaussian

7.0.19 compute_grid_coord_polar (Source File: compute_grid_coord_polar.F90)

REVISION HISTORY:

04-10-96 Mark Iredell; Initial Specification
07-15-05 Sujay Kumar; Modified version with floating point arithmetic.

INTERFACE:

```

subroutine compute_grid_coord_polar(gridDesc,npts,fill,xpts,ypts,&
rlon,rlat,nret)

implicit none

```

ARGUMENTS:

```

real          :: gridDesc(50)
integer        :: npts
real          :: fill
real          :: xpts(npts),ypts(npts)
real          :: rlat(npts)
real          :: rlon(npts)
integer        :: nret

```

DESCRIPTION:

This subroutine computes the grid coordinates of the specified domain for a polar stereographic projection. This routine is based on the grid decoding routines in the NCEP interoplation package.

gridDesc grid description parameters

npts integer maximum number of coordinates

fill fill value to set invalid output data

xpts output grid x point coordinates

ypts output grid y point coordinates

rlat input latitudes in degrees

rlon input longitudes in degrees

The routines invoked are:

latlonTopolar (7.0.22)

Converts lat, lons to polar coordinates

ROUTINE : gausslat

REVISION HISTORY:

```

04-16-92 Mark Iredell; Initial Specification
10-20-97 Mark Iredell; Increased precision
05-14-02 Urzula Jambor; Reduced limit of eps from e-12 to e-7

```

INTERFACE:

```

subroutine gausslat(jmax,slat,wlat)
implicit none

```

ARGUMENTS:

```

integer        :: jmax
real          :: slat(jmax)
real          :: wlat(jmax)

```

DESCRIPTION:

This subroutine computes gaussian latitudes Computes cosines of colatitude and gaussian weights on the gaussian latitudes. the gaussian latitudes are at the zeroes of the legendre polynomial of the given order. The arguments are:

jmax input number of latitudes

slat cosines of colatitude

wlat gaussian weights

7.0.20 get_field_pos (Source File: **get_fieldpos.F90**)

REVISION HISTORY:

04-10-96 Mark Iredell; Initial Specification

03-11-96 Mark Iredell; Allowed hemispheric grids to wrap over one pole

05-27-04 Sujay Kumar; Modified code with floating point arithmetic

INTERFACE:

```
function get_fieldpos(i,j,gridDesc) result(field_pos)
implicit none
```

ARGUMENTS:

```
integer      :: field_pos
real        :: gridDesc(50)
integer      :: i,j
```

DESCRIPTION:

This subprogram returns the field position for a given grid point based on the input grid definition. The arguments are:

i integer x grid point

j integer y grid point

gridDesc grid description parameters

field_pos integer position in grid field to locate grid point

7.0.21 polfixs (Source File: **polfixs.F90**)

REVISION HISTORY:

04-10-96 Mark Iredell; Initial Specification

INTERFACE:

```
subroutine polfixs(nm,nx,km,rlat,rlon,ib,lo,go)
implicit none
```

ARGUMENTS:

```
integer      :: nm
integer      :: nx
integer      :: km
real         :: rlat(nm)
real         :: rlon(nm)
integer      :: ib(km)
logical*1    :: lo(nx,km)
real         :: go(nx,km)
```

DESCRIPTION:

This subroutine averages multiple pole scalar values on a latitude/longitude grid. bitmaps may be averaged too.

The arguments are:

nm number of grid points
nx leading dimension of fields
rlat latitudes in degrees
rlon longitudes in degrees
ib integer bitmap flags
lo logical bitmaps
go returned scalar value

7.0.22 latlonTopolar (Source File: latlonTopolar.F90)

REVISION HISTORY:

```
86-07-17 MCDONELL, J.
88-06-07 R.E.JONES   CLEAN UP CODE, TAKE OUT GOTO, USE THEN, ELSE
89-11-02 R.E.JONES   CHANGE TO CRAY CFT77 FORTRAN
05-27-04 Sujay Kumar Incorporated in LIS
```

INTERFACE:

```
subroutine latlontopolar(alat,along,xmeshl,orient,xi,xj)
```

```
implicit none
```

ARGUMENTS:

```
real      :: alat
real      :: along
real      :: xmeshl
real      :: orient
real      :: xi
real      :: xj
```

DESCRIPTION:

Converts the coordinates of a location on earth from the natural coordinate system of latitude/longitude to the grid (i,j) coordinate system overlaid on a polar stereographic map projection true at 60 degrees n or s latitude.

The arguments are:

alat latitude in degrees
along longitude in degrees
xmeshl mesh length of grid in km at 60deg lat (j0 if sh)
orient orientation west longitude of the grid
xi I of the point relative to North or South pole
xj J of the point relative to North or South pole

7.0.23 polarToLatLon (Source File: polarToLatLon.F90)

!REVISION HISTORY 86-07-17 R.E.JONES 89-11-01 R.E.JONES CHANGE TO CRAY CFT77 FORTRAN
05-27-04 Sujay Kumar Incorporated in LIS

INTERFACE:

```
subroutine polarToLatLon(xi,xj,xmeshl,orient,alat,along)
implicit none
```

ARGUMENTS:

```
real    :: xi
real    :: xj
real    :: xmeshl
real    :: orient
real    :: alat
real    :: along
```

DESCRIPTION:

Converts the coordinates of a location from the grid(i,j) coordinate system overlaid on the polar stereographic map projection true at 60 degrees N or S latitude to the natural coordinate system of latitude/longitude on the earth.

The arguments are:

xi I of the point relative to North or South pole
xj J of the point relative to North or South pole
xmeshl mesh length of grid in km at 60deg lat (j0 if sh)
orient orientation west longitude of the grid
alat latitude in degrees
along longitude in degrees

7.0.24 lltops (Source File: lltops.F90)

REVISION HISTORY:

```
05 aug 1998 initial version.....ssgt mccormick/dnxm
10 aug 1999 ported to ibm sp-2. added intent attributes to
           arguments.....mr gayno/dnxm
29 oct 2005; Sujay Kumar; Incorporated into LIS
```

INTERFACE:

```
subroutine lltops( pose, rlat, rlon, mesh, hemi, ri, rj )  
  implicit none
```

ARGUMENTS:

```
integer,      intent(in)    :: pose  
real,         intent(in)    :: rlat  
real,         intent(in)    :: rlon  
integer,      intent(in)    :: mesh  
integer,      intent(out)   :: hemi  
real,         intent(out)   :: ri  
real,         intent(out)   :: rj
```

DESCRIPTION:

converts lat/lon to polar stereographic grid i/j points
method

calculate map related constants/factors.
adjust the longitude according to the user specified sign convention.
convert longitude to radians.
calculate the distance from the pole to the point in radians.
calculate the center point coordinate on the grid.
calculate the grid coordinates for the point.

The arguments are:

pose longitude increment orientation flag (1-positive east, 0-positive west)
rlat input latitude in degrees
rlon input longitude in degrees
mesh mesh factor
hemi index of the hemisphere (1-nh, 2-sh)
ri i-coordinate
rj j-coordinate

7.0.25 pstoll (Source File: pstoll.F90)

REVISION HISTORY:

```
05 aug 1998 initial version.....ssgt mccormick/dnxm
10 aug 1999 ported to ibm sp2. added intent attributes to
arguments.....mr gayno/dnxm
25 jul 2005 Adopted in LIS
```

INTERFACE:

```
subroutine pstoll( hemi, pose, ri, rj, mesh, rlat, rlon )
implicit none
```

ARGUMENTS:

integer,	intent(in)	:: hemi
integer,	intent(in)	:: pose
real,	intent(in)	:: ri
real,	intent(in)	:: rj
integer,	intent(in)	:: mesh
real,	intent(out)	:: rlat
real,	intent(out)	:: rlon

DESCRIPTION:

converts polar stereographic grid i/j points to lat/lon
method:

initialize grid specific constants.
calculate distance between input point and center point of grid.
using this distance, calculate latitude and longitude using trigonometry.
adjust sign of longitude according to user preference.

The arguments are:

hemi index of the hemisphere (1-nh, 2-sh)
pose longitude increment orientation flag (1-positive east, 0-positive west)
ri i-coordinate
rj j-coordinate
mesh mesh factor
rlat output latitude in degrees
rlon output longitude in degrees

7.1 Fortran: Module Interface map_utils (Source File: map_utils.F90)

Module that defines constants, data structures, and subroutines used to convert grid indices to lat/lon and vice versa. (This module is adopted from the Weather Research and Forecasting (WRF) model Standard Initialization(SI) program.

7.1.1 Supported Projections

Cylindrical Lat/Lon (code = PROJ_LATLON)
Mercator (code = PROJ_MERC)
Lambert Conformal (code = PROJ_LC)
Polar Stereographic (code = PROJ_PS)

7.1.2 Remarks

The routines contained within were adapted from routines obtained from the NCEP w3 library. The original NCEP routines were less flexible (e.g., polar-stereo routines only supported truelat of 60N/60S) than what we needed, so modifications based on equations in Hoke, Hayes, and Renninger (AFGWC/TN/79-003) were added to improve the flexibility. Additionally, coding was improved to F90 standards and the routines were combined into this module.

7.1.3 Assumptions

Grid Definition: For mercator, lambert conformal, and polar-stereographic projections, the routines within assume the following:

- Grid is dimensioned (i,j) where i is the East-West direction, positive toward the east, and j is the north-south direction, positive toward the north.
- Origin is at (1,1) and is located at the southwest corner, regardless of hemisphere.
- Grid spacing (dx) is always positive.
- Values of true latitudes must be positive for NH domains and negative for SH domains.

For the latlon projection, the grid origin may be at any of the corners, and the deltalat and deltalon values can be signed to account for this using the following convention:

Origin Location	Deltalat Sign	Deltalon Sign
SW Corner	+	+
NE Corner	-	-
NW Corner	-	+
SE Corner	+	-

7.1.4 Data Definitions:

- Any arguments that are a latitude value are expressed in degrees north with a valid range of -90 \leq 90
- Any arguments that are a longitude value are expressed in degrees east with a valid range of -180 \leq 180.
- Distances are in meters and are always positive.
- The standard longitude (stdlon) is defined as the longitude line which is parallel to the y-axis (j-direction), along which latitude increases (NOT the absolute value of latitude, but the actual latitude, such that latitude increases continuously from the south pole to the north pole) as j increases.
- One true latitude value is required for polar-stereographic and mercator projections, and defines at which latitude the grid spacing is true. For lambert conformal, two true latitude values must be specified, but may be set equal to each other to specify a tangent projection instead of a secant projection.

7.1.5 Usage

To use the routines in this module, the calling routines must have the following statement at the beginning of its declaration block: `USE map_utils`

The use of the module not only provides access to the necessary routines, but also defines a structure of `TYPE (proj_info)` that can be used to declare a variable of the same type to hold your map projection information. It also defines some integer parameters that contain the projection codes so one only has to use those variable names rather than remembering the acutal code when using them. The basic steps are as follows:

- Ensure the `USE map_utils` is in your declarations.
- Declare the projection information structure as `type(proj_info): TYPE(proj_info) :: proj`
- Populate your structure by calling the `map_set` routine:
`CALL map_set(code,lat1,lon1,dx,stdlon,truelat1,truelat2,nx,ny,proj)` where: code (input) = one of PROJ_LATLON, PROJ_MERC, PROJ_LC, or PROJ_PS
lat1 (input) = Latitude of grid origin point (i,j)=(1,1) (see assumptions!)
lon1 (input) = Longitude of grid origin
dx (input) = grid spacing in meters (ignored for LATLON projections)
stdlon (input) = Standard longitude for PROJ_PS and PROJ_LC, deltalon (see assumptions) for PROJ_LATLON, ignored for PROJ_MERC
truelat1 (input) = 1st true latitude for PROJ_PS, PROJ_LC, and PROJ_MERC, deltalat (see assumptions) for PROJ_LATLON
truelat2 (input) = 2nd true latitude for PROJ_LC, ignored for all others.
nx = number of points in east-west direction
ny = number of points in north-south direction
proj (output) = The structure of `type (proj_info)` that will be fully populated after this call

- Now that the proj structure is populated, you may call any of the following routines:

```
latlon_to_ij(proj, lat, lon, i, j)
ij_to_latlon(proj, i, j, lat, lon)
truewind_to_gridwind(lon, proj, ugrid, vgrid, utrue, vtrue)
gridwind_to_truewind(lon, proj, utrue, vtrue, ugrid, vgrid)
compare_projections(proj1, proj2, same_proj)
```

It is incumbent upon the calling routine to determine whether or not the values returned are within your domain bounds. All values of i, j, lat, and lon are REAL values.

References

Hoke, Hayes, and Renninger, "Map Preojections and Grid Systems for Meteorological Applications." AFGWC/TN-79/003(Rev), Air Weather Service, 1985.

NCAR MM5v3 Modeling System, REGRIDDER program, module_first_guess_map.F NCEP routines w3fb06, w3fb07, w3fb08, w3fb09, w3fb11, w3fb12

REVISION HISTORY:

```
27 Mar 2001 - Original Version
               Brent L. Shaw, NOAA/FSL (CSU/CIRA)
02 Apr 2001 - Added routines to rotate winds from true to grid
               and vice versa.
               Brent L. Shaw, NOAA/FSL (CSU/CIRA)
09 Apr 2001 - Added compare\_projections routine to compare two
               sets of projection parameters.
```

```
implicit none

! mean earth radius in m.  the value below is consistent
! with nceps routines and grids.
real, public, parameter :: earth_radius_m = 6371200.

! define public parameters
mp
  INTEGER, PUBLIC, PARAMETER :: PROJ_ROTLAT = 203
mp
```

PUBLIC MEMBER FUNCTIONS:

```
public :: map_init      !initializes map projection structure
public :: map_set        !completes the initializations
public :: ij_to_latlon  !converts ijs to lat lon values
public :: latlon_to_ij   !converts lat lon values to ijs.
```

PUBLIC TYPES:

```
! projection codes for proj_info structure:
integer, public, parameter :: proj_latlon = 0
integer, public, parameter :: proj_merc = 1
integer, public, parameter :: proj_lc = 3
integer, public, parameter :: proj_ps = 5
```

7.1.6 map_init (Source File: map_utils.F90)

INTERFACE:

```
subroutine map_init(proj)
  implicit none
```

ARGUMENTS:

```
  type(proj_info), intent(inout) :: proj
```

DESCRIPTION:

Initializes the map projection structure to missing values
The arguments are:

proj data structure containing the map projection information

7.1.7 map_set (Source File: map_utils.F90)

INTERFACE:

```

subroutine map_set(proj_code,lat1,lon1,dx,stdlon,truelat1,truelat2, &
                   idim,jdim,proj)
implicit none

```

ARGUMENTS:

integer, intent(in)	:: proj_code
real, intent(in)	:: lat1
real, intent(in)	:: lon1
real, intent(in)	:: dx
real, intent(in)	:: stdlon
real, intent(in)	:: truelat1
real, intent(in)	:: truelat2
integer, intent(in)	:: idim
integer, intent(in)	:: jdim
type(proj_info), intent(out)	:: proj

DESCRIPTION:

Given a partially filled `proj_info` structure, this routine computes `polei`, `polej`, `rsw`, and `cone` (if LC projection) to complete the structure. This allows us to eliminate redundant calculations when calling the coordinate conversion routines multiple times for the same map. This will generally be the first routine called when a user wants to be able to use the coordinate conversion routines, and it will call the appropriate subroutines based on the `proj%code` which indicates which projection type this is.

The arguments are:

- proj_code** code to indicate the type of projection
- lat1** latitude of origin
- lon1** longitude of origin
- dx** grid spacing in meters
- stdlon** standard longitude
- truelat1** 1st true latitude
- truelat2** 2nd true latitude
- idim** number of points in the East-West direction
- jdim** number of points in the North-South direction
- proj** data structure containing the map projection information

7.1.8 latlon_to_ij (Source File: map_utils.F90)

INTERFACE:

```

subroutine latlon_to_ij(proj, lat, lon, i, j)
implicit none

```

ARGUMENTS:

```

type(proj_info), intent(in)      :: proj
real, intent(in)                 :: lat
real, intent(in)                 :: lon
real, intent(out)                :: i
real, intent(out)                :: j

```

DESCRIPTION:

Converts input lat/lon values to the cartesian (i,j) value for the given projection.
The arguments are:

proj data structure containing the map projection information
lat input latitude
lon input longitude
i output i value
j output j value

7.1.9 ij_to_latlon (Source File: map_utils.F90)

INTERFACE:

```

subroutine ij_to_latlon(proj, i, j, lat, lon)
  implicit none

```

ARGUMENTS:

```

type(proj_info),intent(in)      :: proj
real, intent(in)                 :: i
real, intent(in)                 :: j
real, intent(out)                :: lat
real, intent(out)                :: lon

```

DESCRIPTION:

Computes geographical latitude and longitude for a given (i,j) point in a grid with a projection of proj
The arguments are:

proj data structure containing the map projection information
i input i value
j input j value
lat output latitude
lon output longitude

7.1.10 compute_stnwts (Source File: compute_stnwts.F90)

REVISION HISTORY:

07-15-05 Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine compute_stnwts(nstns, gridDesc,stnlat, stnlon,&
    npts, stnwt)

    implicit none
```

ARGUMENTS:

```
integer      :: nstns
real        :: gridDesc(50)
real        :: stnlat(nstns)
real        :: stnlon(nstns)
integer      :: npts
real        :: stnwt(npts,nstns)
```

DESCRIPTION:

This routine compute the interpolation weights to be applied to a network of stations, to generate a gridded field from observations from stations. It simply uses an inverse distance weighting algorithm to compute the relative weights of each station.

nstns number of stations used in interpolation

gridDesc grid description parameters

stnlat station latitudes in degrees

stnlon station longitudes in degrees

npts integer maximum number of coordinates

stnwt interpolation weights of stations with respect to the each point

The routines invoked are:

compute_earth_coord (7.0.8)
computes the earth coordinates

7.1.11 interp_stndata (Source File: interp_stndata.F90)

REVISION HISTORY:

08 Dec 04 Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine interp_stndata(wt,vari,varo,npts,nstns)
    implicit none
```

ARGUMENTS:

```
integer :: nstns,npts
real    :: wt(npts,nstns)
real    :: vari(nstns)
real    :: varo(npts)
```

DESCRIPTION:

This subroutine interpolates station data into a gridded set. Currently works only on a lat/lon grid
The arguments are:

nstns number of stations
npts number of points in the output field
vari input variable (array of observations from the stations)
varo output variable (gridded data on the output field)
wt interpolation weights

7.1.12 normalize_stnwts (Source File: normalize_stnwts.F90)

REVISION HISTORY:

08Dec04 Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine normalize_stnwts(stndata,nstns,npts,undef,wt)
  implicit none
```

ARGUMENTS:

```
integer :: nstns
integer :: npts
real   :: stndata(nstns)
real   :: wt(npts,nstns)
real   :: undef
```

DESCRIPTION:

This subroutine normalizes the interpolation weights to convert station data into a gridded set.
The arguments are:

nstns number of stations used in interpolation
npts integer maximum number of coordinates
stndata input data (station observations)
undef undefined value used in observations
wt interpolation weights

Part VI

User-defined extensible components in LIS

8 User-defined extensible components in LIS

The following sections describe the details of user-defined extensible components in LIS. These include the following:

Running Modes

Domains

Land Surface Parameters

Meteorological forcing analyses

Supplemental forcing products

Land Surface Models

Part VII

Running Modes in LIS

9 Running Modes in LIS

This section contains the implementations of various running modes in LIS. The current implementation includes a retrospective mode and the cycling mode used in the AGRMET system. The interfaces are expected to be extended for other running modes such as coupled, parameter estimation and forecast.

10 Retrospective Running Mode

This is a typical running mode used in LIS to perform retrospective simulations. The mode assumes that all the meteorological forcing analyses required for the specified time period is archived for the simulation.

10.1 Fortran: Module Interface retrospective_runMod (Source File: retrospective_runMod.F90)

This module contains the definition of the functions used for LIS initialization, execution, and finalization for a retrospective runmode in LIS.

REVISION HISTORY:

21Oct05 Sujay Kumar Initial Specification

implicit none

PUBLIC MEMBER FUNCTIONS:

```
public :: lis_init_retrospective !init method for retrospective mode
public :: lis_run_retrospective !run method for retrospective mode
public :: lis_final_retrospective !finalize method for retrospective mode
```

10.1.1 lis_init_retrospective (Source File: retrospective_runMod.F90)

INTERFACE:

```
subroutine lis_init_retrospective
```

USES:

```
use lisdrv_module
use domain_module
use lsm_module
use baseforcing_module
use suppforcing_module
use dataassim_module
use param_module
use spmdMod
```

DESCRIPTION:

This is the initialize method for LIS in a retrospective running mode. The following calls are invoked from this method.

LIS_domain_init (5.8.2)
initialize the LIS domains

LIS_param_init (5.13.2)
initialize parameters

LIS_lsm_init (5.9.3)
initialize the land surface model.

LIS_baseforcing_init (5.10.2)
initialize the base forcing

LIS_suppforcing_init (5.11.2)
initialize the supplemental forcing

LIS_setuplsm (5.9.4)
complete the LSM setups

LIS_readrestart (5.9.6)
read the restart files

10.1.2 lis_run_retrospective (Source File: retrospective_runMod.F90)

INTERFACE:

```
subroutine lis_run_retrospective
```

USES:

```
use lisdrv_module
use lsm_module
use param_module
use baseforcing_module
use suppforcing_module
```

DESCRIPTION:

This is the run method for LIS in a retrospective running mode. The following calls are invoked from this method.

LIS_endofrun (5.1.6)
check to see if the end of simulation has reached

LIS_ticktime (5.1.5)
advance model clock

LIS_timeToRunNest (5.1.7)
check to see if the current nest needs to be run.

LIS_setDynparams (5.13.3)
set the time dependent parameters

LIS_get_base_forcing (5.10.3)
retrieve the base forcing

LIS_get_supp_forcing (5.11.3)
retrieve the supplemental forcing

LIS_force2tile (5.9.2)
transfer forcing to model tiles

LIS_lsm_main (5.9.5)
run the land surface model

LIS_lsm_output (5.9.7)
write model output

LIS_writerestart (5.9.11)
write model restart files

10.1.3 lis_final_retrospective (Source File: retrospective_runMod.F90)

INTERFACE:

```
subroutine lis_final_retrospective
```

USES:

```
use lisdrv_module, only : lis_finalize
use lsm_module
use param_module
use baseforcing_module
use suppforcing_module
use dataassim_module
```

DESCRIPTION:

This is the finalize method for LIS in a retrospective running mode. The following calls are invoked from this method.

LIS_finalize (5.1.8)
cleanup LIS generic structures

LIS_lsm_finalize (5.9.13)
cleanup land surface model specific structures

LIS_param_finalize (5.13.4)
cleanup parameter specific structures

LIS_baseforcing_finalize (5.10.4)
cleanup baseforcing specific structures

LIS_suppforcing_finalize (5.11.4)
cleanup supplemental forcing specific structures

LIS_dataassim_finalize (5.12.3)
cleanup data assimilation specific structures

11 AGRMET Running Mode

This is a running mode customized to simulate the various cyclings used in the AGRMET model.

11.1 Fortran: Module Interface agrmet_runMod (Source File: agrmet_runMod.F90)

This module contains the definition of the functions used for LIS initialization, execution, and finalization for the forecast runmode used by AGRMET in LIS.

REVISION HISTORY:

21Oct05 Sujay Kumar Initial Specification

implicit none

PUBLIC MEMBER FUNCTIONS:

```
public :: lis_init_agrmet !init method for AFWA forecast mode
public :: lis_run_agrmet !run method for AFWA forecast mode
public :: lis_final_agrmet !finalize method for AFWA forecast mode
```

11.1.1 lis_init_agrmet (Source File: agrmet_runMod.F90)

INTERFACE:

```
subroutine lis_init_agrmet
```

USES:

```
use lisdrv_module
use domain_module
use lsm_module
use baseforcing_module
use suppforcing_module
use dataassim_module
use param_module
use spmdMod
```

DESCRIPTION:

This is the initialize method for LIS in the AGRMET running mode. The following calls are invoked from this method.

LIS_domain_init (5.8.2)
initialize the LIS domains

LIS_param_init (5.13.2)
initialize parameters

LIS_lsm_init (5.9.3)
initialize the land surface model.

LIS_baseforcing_init (5.10.2)
initialize the base forcing

LIS_suppforcing_init (5.11.2)
initialize the supplemental forcing

LIS_setuplsm (5.9.4)
complete the LSM setups

LIS_readrestart (5.9.6)
read the restart files

11.1.2 lis_run_agrmet (Source File: agrmet_runMod.F90)

INTERFACE:

```
subroutine lis_run_agrmet
```

USES:

```
use lisdrv_module
use lsm_module
use param_module
use baseforcing_module
use suppforcing_module
use lis_logmod, only : logunit
```

DESCRIPTION:

This is the run method for LIS in the AGRMET running mode. The following calls are invoked from this method.

LIS_endofrun (5.1.6)
check to see if the end of simulation has reached

LIS_ticktime (5.1.5)
advance model clock

LIS_timeToRunNest (5.1.7)
check to see if the current nest needs to be run.

LIS_setDynparams (5.13.3)
set the time dependent parameters

LIS_get_base_forcing (5.10.3)
retrieve the base forcing

LIS_get_supp_forcing (5.11.3)
retrieve the supplemental forcing

LIS_force2tile (5.9.2)
transfer forcing to model tiles

LIS_lsm_main (5.9.5)
run the land surface model

LIS_lsm_output (5.9.7)
write model output

LIS_writerestart (5.9.11)
write model restart files

11.1.3 lis_final_agrmet (Source File: agrmet_runMod.F90)

INTERFACE:

```
subroutine lis_final_agrmet
```

USES:

```
use lisdrv_module, only : lis_finalize
use lsm_module
use param_module
use baseforcing_module
use suppforcing_module
use dataassim_module
```

DESCRIPTION:

This is the finalize method for LIS in the AGRMET running mode. The following calls are invoked from this method.

LIS_finalize (5.1.8)

cleanup LIS generic structures

LIS_lsm_finalize (5.9.13)

cleanup land surface model specific structures

LIS_param_finalize (5.13.4)

cleanup parameter specific structures

LIS_baseforcing_finalize (5.10.4)

cleanup baseforcing specific structures

LIS_suppforcing_finalize (5.11.4)

cleanup supplemental forcing specific structures

LIS_dataassim_finalize (5.12.3)

cleanup data assimilation specific structures

Part VIII

Domain implementations in LIS

12 Domain implementations in LIS

This section contains the implementations of interfaces to incorporate a new domain in LIS. Currently LIS supports operations on a lat/lon, polar stereographic, lambert conformal, mercator, and GSWP grids.

13 Lat/Lon domain

This section contains the interface implementations for a LIS instance using a lat/lon projection with SW to NE data ordering

13.0.4 createtiles_latlon (Source File: `createtiles_latlon.F90`)

REVISION HISTORY:

```
1 Oct 1999: Jared Entin; Initial code
15 Oct 1999: Paul Houser; Major F90 and major structure revision
3 Jan 2000: Minor T=0 bug fix, should have no effect on output
8 Mar 2000: Brian Cosgrove; Initialized FGRD to 0 For Dec Alpha Runs
22 Aug 2000: Brian Cosgrove; Altered code for US/Mexico/Canada Mask
04 Feb 2001: Jon Gottschalck; Added option to read and use Koster tile space
17 Oct 2003: Sujay Kumar ; Initial version of subsetting code
18 Feb 2006: Sujay Kumar ; Added nesting options
```

INTERFACE:

```
subroutine createtiles_latlon()
```

USES:

```
use lisdrv_module, only: lis,lisdom
use grid_module
use spmdMod
use lis_logmod, only :logunit
```

DESCRIPTION:

The code in this routine creates the grid and tile spaces in a lat/lon grid projection, with a SW to NE data ordering. The routine first reads the landmask and landcover data. The topography data is also read in depending on the options specified at runtime. The landmask is used to create the grid space, ignoring the water points. The vegetation distribution from the landcover map is used to create subgrid tiling, based on the specified maximum number of tiles and minimum cutoff percentage.

The routines invoked are:

readlandmask (5.33.4)

calls the generic routine in the registry to read the landmask

readlandcover (5.33.2)

calls the generic routine in the registry to read the landcover data

readelev (5.40.2)

calls the generic routine in the registry to read the elevation data

readslope (5.40.4)

calls the generic routine in the registry to read the slope data

readaspect (5.40.6)

calls the generic routine in the registry to read the aspect data

readcurv (5.40.8)

calls the generic routine in the registry to read the curvature data

calculate_domveg (13.0.5)

computes the dominant vegetation distribution for subgrid tiling

create_vegtilespace_latlon (13.0.6)

computes the grid and tile spaces based on the landmask, landcover, and topography datasets in a latlon projection

13.0.5 calculate_domveg (Source File: calculate_domveg.F90)

REVISION HISTORY:

09 Sept 2004: Sujay Kumar ; Initial version

INTERFACE:

```
subroutine calculate_domveg(n, fgrd)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod, only      : iam

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real :: fgrd(lis%lnc(n), lis%lnr(n), lis%nt)
```

DESCRIPTION:

This primary goal of this routine is to determine the percentages of dominant vegetation for subgrid tiling. The routine uses the distribution of vegetation within a grid cell and imposes the specified cutoff percentage and the maximum number of veg types allowed to create the number of tiles.

The arguments are:

n index of the nest

fgrd fraction of grid covered by each vegetation type

13.0.6 create_vegtilespace_latlon (Source File: create_vegtilespace_latlon.F90)

REVISION HISTORY:

09 Sept 2004: Sujay Kumar ; Initial version

INTERFACE:

```
subroutine create_vegtilespace_latlon(n, fgrd, localmask)
      slope, aspect, curv)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real :: fgrd(lis%lnc(n), lis%lnr(n), lis%nt)
real :: localmask(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This routine computes the grid and tile spaces based on the input distribution of vegetation and landmask. The routine also computes the following structures necessary for domain decomposition and other parallel routines.

ntiles_pergrid : Number of tiles per each grid

s_tid : Index of the first tile in each grid cell

The arguments are:

n index of the nest

fgrd fraction of grid covered by each vegetation type

localmask landmask for the region of interest

13.0.7 readinput_latlon (Source File: readinput_latlon.F90)

REVISION HISTORY:

```
15 Oct 1999: Paul Houser; Initial code
11 Apr 2000: Brian Cosgrove; Added Elevation correction and Forcing
              Mask read statements
14 Nov 2003: Sujay Kumar; Modified card file that includes regional
              modeling options
```

INTERFACE:

```
subroutine readinput_latlon
```

USES:

```
use lisdrv_module, only : lis, config_lis
use spmdMod
use lis_logmod, only : logunit
use LIS_ConfigMod
```

DESCRIPTION:

This routine reads the options specified in the LIS configuration file to determine the region of interest in the LIS simulation, in a lat/lon projection. The routine reads the extents of the running domain. The land surface parameters used for the simulation are read from a file that spans the area of interest. The domain specifications of the parameter maps are also read in by this routine. Based on the number of processors and the processor layout specified, this routine also performs the domain decomposition.
The routines invoked are:

LIS_ConfigFindLabel (5.7.8)

find the specified label to read the attribute

LIS_ConfigGetAttribute (5.7.5)

read the specified attribute

listask_for_point (5.41.8)

routine to perform domain decomposition

14 Mercator domain

This section contains the interface implementations for a LIS instance using a mercator projection with SW to NE data ordering

14.0.8 createtiles_merc (Source File: **createtiles_merc.F90**)

REVISION HISTORY:

17 Jul 2005: Sujay Kumar ; Initial specification

INTERFACE:

```
subroutine createtiles_merc()
```

USES:

```
use lisdrv_module,only : lis,lisdom
use spmdMod, only : iam
implicit none
```

DESCRIPTION:

The code in this routine creates the grid and tile spaces in a mercator grid projection, with a SW to NE data ordering. The routine first reads the landmask and landcover data. The topography data is also read in depending on the options specified at runtime. The landmask is used to create the grid space, ignoring the water points. The vegetation distribution from the landcover map is used to create subgrid tiling, based on the specified maximum number of tiles and minimum cutoff percentage.

The routines invoked are:

readlandmask (5.33.4)

calls the generic routine in the registry to read the landmask

readlandcover (5.33.2)

calls the generic routine in the registry to read the landcover data

readelev (5.40.2)

calls the generic routine in the registry to read the elevation data

readslope (5.40.4)

calls the generic routine in the registry to read the slope data

readaspect (5.40.6)

calls the generic routine in the registry to read the aspect data

readcurv (5.40.8)

calls the generic routine in the registry to read the curvature data

calculate_domveg (13.0.5)

computes the dominant vegetation distribution for subgrid tiling

create_vegtilespace_merc (14.0.9)

computes the grid and tile spaces based on the landmask, landcover, and topography datasets for the mercator projection

14.0.9 create_vegtilespace_merc (Source File: create_vegtilespace_merc.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar ; Initial version

INTERFACE:

```
subroutine create_vegtilespace_merc(n,fgrd, localmask)
    slope, aspect, curv)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                 :: fgrd(lis%lnc(n), lis%lnr(n), lis%nt)
real                 :: localmask(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This routine computes the grid and tile spaces based on the input distribution of vegetation and landmask. The routine also computes the following structures necessary for domain decomposition and other parallel routines.

ntiles_pergrid : Number of tiles per each grid

s_tid : Index of the first tile in each grid cell

The arguments are:

n index of the nest

fgrd fraction of grid covered by each vegetation type

localmask landmask for the region of interest

14.0.10 readinput_merc (Source File: readinput_merc.F90)

REVISION HISTORY:

15 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine readinput_merc
```

USES:

```
use lisdrv_module, only : lis,lisdom, config_lis
use spmdMod
use lis_logmod, only : logunit
use map_utils
use LIS_ConfigMod
```

DESCRIPTION:

This routine reads the options specified in the LIS configuration file to determine the region of interest in the LIS simulation, in a mercator projection. The routine reads the extents of the running domain. The land surface parameters used for the simulation are read from a file that spans the area of interest. The domain specifications of the parameter maps are also read in by this routine. Based on the number of processors and the processor layout specified, this routine also performs the domain decomposition.
The routines invoked are:

LIS_ConfigFindLabel (5.7.8)

find the specified label to read the attribute

LIS_ConfigGetAttribute (5.7.5)

read the specified attribute

listask_for_point (5.41.8)

routine to perform domain decomposition

15 Lambert Conformal domain

This section contains the interface implementations for a LIS instance using a lambert conformal projection with SW to NE data ordering

15.0.11 createtiles_lambert (Source File: createtiles_lambert.F90)

REVISION HISTORY:

17 Jul 2005: Sujay Kumar ; Initial specification

INTERFACE:

```
subroutine createtiles_lambert()
```

USES:

```
use lisdrv_module,only : lis,lisdom
use spmdMod, only : iam

implicit none
```

DESCRIPTION:

The code in this routine creates the grid and tile spaces in a lambert conformal grid projection, with a SW to NE data ordering. The routine first reads the landmask and landcover data. The topography data is

also read in depending on the options specified at runtime. The landmask is used to create the grid space, ignoring the water points. The vegetation distribution from the landcover map is used to create subgrid tiling, based on the specified maximum number of tiles and minimum cutoff percentage.

The routines invoked are:

readlandmask (5.33.4)

calls the generic routine in the registry to read the landmask

readlandcover (5.33.2)

calls the generic routine in the registry to read the landcover data

readelev (5.40.2)

calls the generic routine in the registry to read the elevation data

readslope (5.40.4)

calls the generic routine in the registry to read the slope data

readaspect (5.40.6)

calls the generic routine in the registry to read the aspect data

readcurv (5.40.8)

calls the generic routine in the registry to read the curvature data

calculate_domveg (13.0.5)

computes the dominant vegetation distribution for subgrid tiling

create_vegitilespace_lambert (15.0.12)

computes the grid and tile spaces based on the landmask, landcover, and topography datasets for lambert conformal projection.

15.0.12 create_vegitilespace_lambert (Source File: create_vegitilespace_lambert.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar ; Initial version

INTERFACE:

```
subroutine create_vegitilespace_lambert(n,fgrd, localmask)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n
real                  :: fgrd(lis%lnc(n), lis%lnr(n), lis%nt)
real                  :: localmask(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This routine computes the grid and tile spaces based on the input distribution of vegetation and landmask. The routine also computes the following structures necessary for domain decomposition and other parallel routines.

ntiles_pergrid : Number of tiles per each grid

s_tid : Index of the first tile in each grid cell

The arguments are:

n index of the nest

fgrd fraction of grid covered by each vegetation type

localmask landmask for the region of interest

15.0.13 readinput_lambert (Source File: readinput_lambert.F90)

REVISION HISTORY:

15Jul2005: Sujay Kumar; Initial Specification

INTERFACE:

subroutine readinput_lambert

USES:

```
use lisdrv_module, only : lis,config_lis, lisdom
use spmdMod
use lis_logmod, only : logunit
use map_utils
use LIS_ConfigMod
implicit none
```

DESCRIPTION:

This routine reads the options specified in the LIS configuration file to determine the region of interest in the LIS simulation, in a lambert conformal projection. The routine reads the extents of the running domain. The land surface parameters used for the simulation are read from a file that spans the area of interest. The domain specifications of the parameter maps are also read in by this routine. Based on the number of processors and the processor layout specified, this routine also performs the domain decomposition.

The routines invoked are:

LIS_ConfigFindLabel (5.7.8)

find the specified label to read the attribute

LIS_ConfigGetAttribute (5.7.5)

read the specified attribute

listask_for_point (5.41.8)

routine to perform domain decomposition

16 Polar stereographic domain

This section contains the interface implementations for a LIS instance using a polar stereographic projection with SW to NE data ordering

16.0.14 createtiles_polar (Source File: createtiles_polar.F90)

REVISION HISTORY:

17Jul2005: Sujay Kumar ; Initial specification

INTERFACE:

```
subroutine createtiles_polar()
```

USES:

```
use lisdrv_module,only : lis, lisdom
use spmdMod, only : iam
```

DESCRIPTION:

The code in this routine creates the grid and tile spaces in a polar stereographic grid projection, with a SW to NE data ordering. The routine first reads the landmask and landcover data. The topography data is also read in depending on the options specified at runtime. The landmask is used to create the grid space, ignoring the water points. The vegetation distribution from the landcover map is used to create subgrid tiling, based on the specified maximum number of tiles and minimum cutoff percentage.

The routines invoked are:

readlandmask (5.33.4)

calls the generic routine in the registry to read the landmask

readlandcover (5.33.2)

calls the generic routine in the registry to read the landcover data

readelev (5.40.2)

calls the generic routine in the registry to read the elevation data

readslope (5.40.4)

calls the generic routine in the registry to read the slope data

readaspect (5.40.6)

calls the generic routine in the registry to read the aspect data

readcurv (5.40.8)

calls the generic routine in the registry to read the curvature data

calculate_domveg (13.0.5)

computes the dominant vegetation distribution for subgrid tiling

create_vegtilespace_polar (16.0.15)

computes the grid and tile spaces based on the landmask, landcover, and topography datasets in a polar stereographic projection

16.0.15 create_vegtilespace_polar (Source File: create_vegtilespace_polar.F90)

REVISION HISTORY:

09 Sept 2004: Sujay Kumar ; Initial version

INTERFACE:

```

subroutine create_vegitilespace_polar(n, fgrd, localmask)
    slope, aspect, curv)

USES:
use lisdrv_module, only : lis, lisdom
use spmdMod
use lis_logmod, only : logunit
use map_utils

implicit none

integer, intent(in) :: n
real             :: fgrd(lis%lnc(n), lis%lnr(n), lis%nt)
real             :: localmask(lis%lnc(n), lis%lnr(n))

```

DESCRIPTION:

This routine computes the grid and tile spaces based on the input distribution of vegetation and landmask. The routine also computes the following structures necessary for domain decomposition and other parallel routines.

ntiles_pergrid : Number of tiles per each grid
s_tid : Index of the first tile in each grid cell

The arguments are:

n index of the nest
fgrd fraction of grid covered by each vegetation type
localmask landmask for the region of interest

16.0.16 readinput_polar (Source File: readinput_polar.F90)

REVISION HISTORY:

15 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine readinput_polar
```

USES:

```

use lisdrv_module, only : lis,lisdom, config_lis
use spmdMod
use lis_logmod, only : logunit
use map_utils
use LIS_ConfigMod
implicit none

```

DESCRIPTION:

This routine reads the options specified in the LIS configuration file to determine the region of interest in the LIS simulation, in a polar stereographic projection. The routine reads the extents of the running domain. The land surface parameters used for the simulation are read from a file that spans the area of interest. The domain specifications of the parameter maps are also read in by this routine. Based on the number of processors and the processor layout specified, this routine also performs the domain decomposition. The routines invoked are:

LIS_ConfigFindLabel (5.7.8)

find the specified label to read the attribute

LIS_ConfigGetAttribute (5.7.5)

read the specified attribute

listask_for_point (5.41.8)

routine to perform domain decomposition

17 GSWP domain

This section contains the interface implementations for a LIS instance using the domain used in the Global Soil Wetness Project (GSWP).

17.0.17 createtiles_gswp (Source File: **createtiles_gswp.F90**)

REVISION HISTORY:

23 Feb 04, Sujay Kumar : Intial Specification

INTERFACE:

```
subroutine createtiles_gswp()
```

USES:

```
use lisdrv_module, only: lis, lisdom
use lis_logmod, only : logunit
use spmdMod
#if ( defined USE_NETCDF )
use netcdf
#endif
```

DESCRIPTION:

The code in this file implements the grid and tile spaces for the GSWP domain. The routine first reads the landmask and the landcover data. The size of the grid space is determined by ignoring the water points. The GSWP domain does not support subgrid tiling. As a result, the grid and tile spaces are equivalent.

17.0.18 readinput_gswp (Source File: **readinput_gswp.F90**)

REVISION HISTORY:

14 Nov 2003: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine readinput_gswp
```

USES:

```
use lisdrv_module, only : lis, config_lis
use spmdMod
use lis_logmod, only :logunit
use LIS_ConfigMod
```

DESCRIPTION:

This routine reads the domain-related runtime options for GSWP. The GSWP grid is a lat/lon 1deg global grid spanning from 90S to 60N. The extents of the running domain are specified in the LIS configuration file, and read by this routine. The routine also reads the domain extents of the parameter maps. In GSWP, all the parameters are also specified in a lat/lon 1deg global grid.

18 AFWA/AGRMET domain

This section contains the interface implementations for a LIS instance using a lat/lon projection with SW to NE data ordering using the AFWA/AGRMET data.

18.0.19 createtiles_afwa (Source File: `createtiles_afwa.F90`)

REVISION HISTORY:

1 Mar 06, Sujay Kumar : Initial Specification

INTERFACE:

```
subroutine createtiles_afwa()
```

USES:

```
use lisdrv_module, only: lis, lisdom
use lis_logmod, only : logunit
use spmdMod
```

DESCRIPTION:

The code in this file implements the grid and tile spaces for the 1deg lat/lon AFWA domain. The routine first reads the landmask and the landcover data. The size of the grid space is determined by ignoring the water points. The AFWA domain does not support subgrid tiling. As a result, the grid and tile spaces are equivalent.

The routines invoked are:

readlandmask (5.33.4)

calls the generic routine in the registry to read the landmask

readlandcover (5.33.2)

calls the generic routine in the registry to read the landcover data

readslope (5.40.4)

calls the generic routine in the registry to read the slope data

18.0.20 readinput_afwa (Source File: `readinput_afwa.F90`)

REVISION HISTORY:

1 Mar 2006: Sujay Kumar; Modified card file that includes regional modeling options

INTERFACE:

```
subroutine readinput_afwa
```

USES:

```
use lisdrv_module, only : lis, config_lis
use spmdMod
use lis_logmod, only : logunit
use LIS_ConfigMod
```

DESCRIPTION:

This routine reads the options specified in the LIS configuration file to determine the region of interest in the LIS simulation, in a lat/lon projection. The routine reads the extents of the running domain. The land surface parameters used for the simulation are read from a file that spans the area of interest. The domain specifications of the parameter maps are also read in by this routine. Based on the number of processors and the processor layout specified, this routine also performs the domain decomposition. This routine is very similar to the lat/lon processing done in LIS, but has been customized for AFWA benchmarking.

The routines invoked are:

LIS_ConfigFindLabel (5.7.8)

find the specified label to read the attribute

LIS_ConfigGetAttribute (5.7.5)

read the specified attribute

listask_for_point (5.41.8)

routine to perform domain decomposition

Part IX

Land Surface Parameters in LIS

19 Land Surface Parameters in LIS

This section contains the implementations of various land surface parameter maps. The interfaces are primarily geared towards including satellite and observed maps of parameters such as albedo, landcover, greenness, LAI, and soil parameters.

20 Landmask/Landcover Data

This section describes the routines to ingest various sources of landmask/landcover data in different LIS domains

20.0.21 read_latlon_umdmask (Source File: `read_latlon_umdmask.F90`)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_latlon_umdmask(nest, localmask)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: nest
real                 :: localmask(lis%lnc(nest),lis%lnr(nest))
```

DESCRIPTION:

This subroutine reads the UMD landmask data and returns the array in a lat/lon projection.
The arguments are:

n index of nest

localmask landmask for the region of interest

20.0.22 read_latlon_umdlc (Source File: `read_latlon_umdlc.F90`)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_latlon_umdlc(n, fgrd)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real    :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the UMD landcover data and returns the distribution of vegetation in each grid cell, in a lat/lon projection. The data has 13 vegetation types.

The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

20.0.23 read_latlon_usgsmask (Source File: **read_latlon_usgsmask.F90**)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_latlon_usgsmask(nest, localmask)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: nest
real               :: localmask(lis%lnc(nest),lis%lnr(nest))
```

DESCRIPTION:

This subroutine reads the USGS landmask data and returns the array in a lat/lon projection. The arguments are:

n index of nest

localmask landmask for the region of interest

20.0.24 read_latlon_usgslc (Source File: read_latlon_usgslc.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_latlon_usgslc(n, fgrd)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use lis_logmod, only : logunit
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real    :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the USGS landcover data and returns the distribution of vegetation in each grid cell, in a lat/lon projection. The data has 24 vegetation types.

The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

20.0.25 read_lambert_umdmask (Source File: read_lambert_umdmask.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_lambert_umdmask(n,localmask)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod
use lis_logmod, only : logunit
use map_utils
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                :: localmask(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine reads the landmask data and returns the array in a lambert conformal projection.
The arguments are:

n index of nest

localmask landmask for the region of interest

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in lambert conformal projection

20.0.26 read_lambert_umdlc (Source File: read_lambert_umdlc.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_lambert_umdlc(n,fgrd)
```

USES:

```
use lisdrv_module, only : lis,lisdom  
use spmdMod  
use lis_logmod, only :logunit  
use map_utils
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n  
real :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the UMD landcover data and returns the distribution of vegetation in each grid cell, in a lambert conformal projection.

The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in lambert conformal projection

20.0.27 read_lambert_usgsmask (Source File: read_lambert_usgsmask.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_lambert_usgsmask(n,localmask)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                 :: localmask(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine reads the landmask data and returns the array in a lambert conformal projection.
The arguments are:

n index of nest

localmask landmask for the region of interest

The routines invoked are:

ij_to_latlon (7.1.9)
computes the lat lon values in lambert conformal projection

20.0.28 read_lambert_usgslc (Source File: read_lambert_usgslc.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_lambert_usgslc(n,fgrd)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the USGS landcover data and returns the distribution of vegetation in each grid cell, in a lambert conformal projection.

The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in lambert conformal projection

20.0.29 read_merc_umdmask (Source File: `read_merc_umdmask.F90`)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_merc_umdmask(n,localmask)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod
use lis_logmod, only : logunit
use map_utils
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real :: localmask(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine reads the landmask data and returns the array in a mercator projection.
The arguments are:

n index of nest

localmask landmask for the region of interest

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in mercator projection

20.0.30 read_merc_umdlc (Source File: read_merc_umdlc.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_merc_umdlc(n,fgrd)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                 :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the UMD landcover data and returns the distribution of vegetation in each grid cell, in mercator projection.

The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

The routines invoked are:

ij_to_latlon (7.1.9)
computes the lat lon values in mercator projection

20.0.31 read_merc_usgsmask (Source File: read_merc_usgsmask.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_merc_usgsmask(n,localmask)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real :: localmask(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine reads the landmask data and returns the array in a mercator projection.
The arguments are:

n index of nest

localmask landmask for the region of interest

The routines invoked are:

ij_to_latlon (7.1.9)
computes the lat lon values in mercator projection

20.0.32 read_merc_usgslc (Source File: read_merc_usgslc.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_merc_usgslc(n,fgrd)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the USGS landcover data and returns the distribution of vegetation in each grid cell, in mercator projection.

The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

The routines invoked are:

ij_to_latlon (7.1.9)
computes the lat lon values in mercator projection

20.0.33 read_polar_umdmask (Source File: read_polar_umdmask.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_polar_umdmask(n, localmask)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real :: localmask(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine reads the landmask data and returns the array in polar stereographic projection.
The arguments are:

n index of nest

localmask landmask for the region of interest

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in polar stereographic projection

20.0.34 read_polar_umdlc (Source File: read_polar_umdlc.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_polar_umdlc(n,fgrd)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                 :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the UMD landcover data and returns the distribution of vegetation in each grid cell, in a polar stereographic projection.

The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in polar stereographic projection

20.0.35 read_polar_usgsmask (Source File: read_polar_usgsmask.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_polar_usgsmask(n, localmask)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use spmdMod
use lis_logmod, only : logunit
use map_utils
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real                 :: localmask(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine reads the landmask data and returns the array in polar stereographic projection.
The arguments are:

n index of nest

localmask landmask for the region of interest

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in polar stereographic projection

20.0.36 read_polar_usgslc (Source File: read_polar_usgslc.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_polar_usgslc(n,fgrd)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod, only :logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the USGS landcover data and returns the distribution of vegetation in each grid cell, in a polar stereographic projection.

The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in polar stereographic projection

20.0.37 read_afwalc (Source File: read_afwalc.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_afwalc(n, fgrd)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use lis_logmod, only :logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real    :: fgrd(lis%lnc(n),lis%lnr(n),lis%nt)
```

DESCRIPTION:

This subroutine reads the AFWA USGS landcover data and returns the distribution of vegetation in each grid cell, in a lat/lon projection. The data has 24 vegetation types.

The arguments are:

n index of nest

fgrd fraction of grid covered by each vegetation type

20.0.38 read_afwamask (Source File: **read_afwamask.F90**)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_afwamask(nest, localmask)
```

USES:

```
use lisdrv_module, only : lis
use spmdMod
use lis_logmod, only : logunit
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: nest
real               :: localmask(lis%lnc(nest),lis%lnr(nest))
```

DESCRIPTION:

This subroutine reads the AFWA landmask data and returns the array in a lat/lon projection. The arguments are:

n index of nest

localmask landmask for the region of interest

21 Albedo Data

This section describes the routines to ingest various sources of albedo data in different LIS domains

21.0.39 read_gswpalbedo (Source File: **read_gswpalbedo.F90**)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_gswpalbedo(n,mo, array)
```

USES:

```
#if (defined USE_NETCDF)
use netcdf
use spmdMod, only : masterproc
use lisdrv_module, only : lis,lisdom
use lis_logmod, only :lis_log_msg
use gswp_module,   only : getgswp_monindex

implicit none
```

ARGUMENTS:

integer, intent(in)	:: n
integer, intent(in)	:: mo
real, intent(inout)	:: array(lis%lnc(n),lis%lnr(n))

DESCRIPTION:

This subroutine retrieves the albedo climatology for the specified month and returns the values in the GSWP data order

The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved albedo

The routines invoked are:

getgswp_monindex (5.22.1)
computes the GSWP monthly index

21.0.40 read_gswpmxsnoalb (Source File: **read_gswpmxsnoalb.F90**)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_gswpmxsnoalb(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_logmod, only : lis_log_msg

implicit none
```

ARGUMENTS:

integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))

DESCRIPTION:

This subroutine retrieves the maximum snow expected over deep snow to be used in the GSWP data order
The arguments are:

n index of the nest

array output field with the retrieved albedo

21.0.41 read_lcalbedo (Source File: **read_lcalbedo.F90**)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_lcalbedo(n, q, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the albedo climatology for the specified month and returns the values in lambert conformal projection

The arguments are:

n index of the nest

q time index (month or quarter)

array output field with the retrieved albedo

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in lambert conformal projection

21.0.42 read_lcmxsnoalb (Source File: read_lcmxsnoalb.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_lcmxsnoalb(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_fileIOMod, only : lis_open_file
use map_utils
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the maximum snow expected over deep snow and returns the values in lambert conformal projection

The arguments are:

n index of the nest

array output field with the retrieved albedo

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in lambert conformal projection

21.0.43 read_llalbedo (Source File: read_llalbedo.F90)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_llalbedo(n, q, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis
use lis_fileIOMod, only : lis_open_file, lis_read_file
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the albedo climatology for the specified month and returns the values in a latlon projection

The arguments are:

n index of the nest

q time index (month or quarter)

array output field with the retrieved albedo

21.0.44 read_llmxsnoalb (Source File: read_llmxsnoalb.F90)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_llmxsnoalb(n,array)
```

USES:

```
use lisdrv_module, only : lis
use lis_fileIOMod, only : lis_open_file, lis_read_file
```

21.0.45 read_mercalbedo (Source File: read_mercalbedo.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_mercalbedo(n,q, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the albedo climatology for the specified month and returns the values in the mercator projection

The arguments are:

n index of the nest

q time index (month or quarter)

array output field with the retrieved albedo

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in the mercator projection

21.0.46 read_mercmxsnoalb (Source File: **read_mercmxsnoalb.F90**)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_mercmxsnoalb(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_fileIOMod, only : lis_open_file
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the maximum snow expected over deep snow and returns the values in mercator projection

The arguments are:

n index of the nest

array output field with the retrieved albedo

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in mercator projection

21.0.47 read_psalbedo (Source File: read_psalbedo.F90)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_psalbedo(n, q, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use lis_logmod, only : logunit
use map_utils
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the albedo climatology for the specified month and returns the values in the polar stereographic projection

The arguments are:

n index of the nest

q time index (month or quarter)

array output field with the retrieved albedo

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in the polar stereographic projection

21.0.48 read_psmxsnoalb (Source File: read_psmxsnoalb.F90)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_psmxsnoalb(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_fileIOMod, only : lis_open_file
use map_utils
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n  
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the maximum snow expected over deep snow and returns the values in the polar stereographic projection

The arguments are:

n index of the nest

array output field with the retrieved albedo

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in the polar stereographic projection

21.0.49 read_afwamxsnoalb (Source File: read_afwamxsnoalb.F90)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_afwamxsnoalb(n,array)
```

USES:

```
use lisdrv_module, only : lis  
use lis_fileIOMod, only : lis_open_file, lis_read_file
```

22 Greenness Fraction Data

This section describes the routines to ingest various sources of greenness fraction data in different LIS domains

22.0.50 read_gswpgfrac (Source File: read_gswpgfrac.F90)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_gswpgfrac(n,mo, array)
```

USES:

```
#if (defined USE_NETCDF)  
use netcdf  
use spmdMod, only : masterproc  
use lisdrv_module, only : lis,lisdom  
use gswp_module, only : getgswp_monindex  
use lis_logmod, only : logunit,lis_log_msg  
  
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the greenness fraction climatology for the specified month in the GSWP data order. The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved greenness fraction

The routines invoked are:

getgswp_monindex (5.22.1)

computes the GSWP monthly index

22.0.51 read_lcgfrac (Source File: **read_lcgfrac.F90**)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_lcgfrac(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file, lis_set_filename
use lis_logmod, only : logunit
use map_utils
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the greenness fraction climatology for the specified month and returns the values in the lambert conformal projection

The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved greenness fraction

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in the lambert conformal projection

22.0.52 read_llgfrac (Source File: read_llgfrac.F90)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification
20 Feb 2006: Sujay Kumar; Modified to support nesting

INTERFACE:

```
subroutine read_llgfrac(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis
use lis_fileIOMod, only : lis_set_filename, lis_open_file, lis_read_file
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the greenness fraction climatology for the specified month and returns the values in the latlon projection

The arguments are:

n index of the nest
mo time index (month or quarter)
array output field with the retrieved greenness fraction

22.0.53 read_mercgfrac (Source File: read_mercgfrac.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_mercgfrac(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_set_filename, lis_open_file
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the greenness fraction climatology for the specified month and returns the values in the mercator projection

The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved greenness fraction

The routines invoked are:

ij_to_latlon (7.1.9)
computes the lat lon values in mercator projection

22.0.54 read_psgfrac (Source File: read_psgfrac.F90)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_psgfrac(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file, lis_set_filename
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the greenness fraction climatology for the specified month and returns the values in the polar stereographic projection

The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved greenness fraction

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in the polar stereographic projection

22.0.55 read_afwagfrac (Source File: read_afwagfrac.F90)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification
20 Feb 2006: Sujay Kumar; Modified to support nesting

INTERFACE:

```
subroutine read_afwagfrac(n, mo, array)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis
use lis_fileIOMOD, only : lis_set_filename, lis_open_file, lis_read_file
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: mo
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the AFWA greenness fraction climatology for the specified month and returns the values in the latlon projection

The arguments are:

n index of the nest

mo time index (month or quarter)

array output field with the retrieved greenness fraction

23 Leaf Area Index Data

This section describes the routines to ingest various sources of LAI data in different LIS domains

23.0.56 read_gswplai (Source File: read_gswplai.F90)

REVISION HISTORY:

07 Jul 2005: James Geiger, Initial Specification

INTERFACE:

```
subroutine read_gswplai(n,lai1, lai2, wt1, wt2)
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr
use gswp_module, only : getgswp_monindex
use lis_logmod, only : logunit, lis_log_msg
#if ( defined USE_NETCDF )
use netcdf
#endif

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, dimension(lis%nch(n)), intent(out) :: lai1, lai2
real, intent(out) :: wt1, wt2
```

DESCRIPTION:

This program reads in GSWP-2 LAI data
*****Need to be modified *****

23.0.57 read_gswpsai (Source File: read_gswpsai.F90)

REVISION HISTORY:

07 Jul 2005: James Geiger, Initial Specification

INTERFACE:

```
subroutine read_gswpsai(n, sai1, sai2, wt1, wt2)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use listime_mgr

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, dimension(lis%nch(n)), intent(out) :: sai1, sai2
real, intent(out) :: wt1, wt2
```

DESCRIPTION:

This program reads in GSWP-2 SAI data
***** Need to be modified *****

23.0.58 read_ll_avhrrlai (Source File: read_ll_avhrrlai.F90)

REVISION HISTORY:

27 Nov 2001: Jon Gottschalck; Initial code
20 Feb 2002: Jon Gottschalck; Modified to use for 1/4 and 2x2.5 using 1/8 degree monthly data
10 Sept 2004: Sujay Kumar, Initial Specification

INTERFACE:

```
subroutine read_ll_avhrrlai(n, lai1, lai2, wt1, wt2)
```

USES:

```
use listime_mgr
use lisdrv_module, only : lis
use filename_mod
use precision
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real(r8)           :: lai1(lis%nch(n)), lai2(lis%nch(n))
real              :: wt1, wt2
```

DESCRIPTION:

This program reads in AVHRR climatology LAI data and reprojects it on a lat/lon grid
***** Need to be modified*****

23.0.59 read_ll_avhrrsai (Source File: read_ll_avhrrsai.F90)

REVISION HISTORY:

27 Nov 2001: Jon Gottschalck; Initial code
20 Feb 2002: Jon Gottschalck; Modified to use for 1/4 and 2x2.5 using 1/8 degree monthly data
01 Oct 2002: Jon Gottschalck; Modified to add MODIS LAI data

INTERFACE:

```
subroutine read_ll_avhrrsai(n, sai1, sai2, wt1, wt2)
```

USES:

```
use listime_mgr
use lisdrv_module, only : lis
use filename_mod
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real :: sai1(lis%nch(n)),sai2(lis%nch(n))
real :: wt1, wt2
```

DESCRIPTION:

This program reads in AVHRR SAI data in the lat/lon projection
***** Need to be modified*****

23.0.60 climatologylai_llread (Source File: climatologylai_llread.F90)

REVISION HISTORY:

```
27 Nov 2001: Jon Gottschalck; Initial code
20 Feb 2002: Jon Gottschalck; Modified to use for 1/4 and 2x2.5 using 1/8 degree monthly data
01 Oct 2002: Jon Gottschalck; Modified to add MODIS LAI data
```

INTERFACE:

```
subroutine climatologylai_llread(n, name11, name12, lai_t1_f, lai_t2_f)
```

USES:

```
use listime_mgr
use lisdrv_module, only : lis, lisdom
use spmdMod, only : iam
use precision
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
character(len=80) :: name11, name12
real(r8) :: lai_t1_f(lis%nch(n))
real(r8) :: lai_t2_f(lis%nch(n))
```

DESCRIPTION:

This program reads in AVHRR LAI data and reprojects it onto a lat/lon projection
***** Need to be modified*****

23.0.61 climatologysai_llread (Source File: climatologysai_llread.F90)

REVISION HISTORY:

```
27 Nov 2001: Jon Gottschalck; Initial code
20 Feb 2002: Jon Gottschalck; Modified to use for 1/4 and 2x2.5 using 1/8 degree monthly data
01 Oct 2002: Jon Gottschalck; Modified to add MODIS LAI data
```

INTERFACE:

```
subroutine climatologysai_llread(n, name15,name16,sai_t1_f,sai_t2_f)
```

USES:

```
use listime_mgr
use lisdrv_module, only : lis, lisdom
use spmdMod, only : iam
use precision
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
character(len=80) :: name15, name16
real(r8) :: sai_t1_f(lis%nch(n))
real(r8) :: sai_t2_f(lis%nch(n))
```

DESCRIPTION:

This program reads in AVHRR SAI data and reprojects it into a lat/lon projection
*****Need to be modified*****

24 Soils

This section describes the routines to ingest various sources of soil parameter data in different LIS domains

24.0.62 read_gswp_bexp (Source File: read_gswp_bexp.F90)

REVISION HISTORY:

12 Sep 2005: James Geiger; Initial Specification

INTERFACE:

```
subroutine read_gswp_bexp(n,array)
```

USES:

```
#if ( defined USE_NETCDF )
  use netcdf
#endif
use lisdrv_module, only : lis, lisdom
use lis_logmod, only : lis_log_msg

implicit none
```

USES:

```
integer, intent(in)      :: n
real, intent(inout)     :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the b parameter data in the GSWP data order
The arguments are:

n index of the nest

array output field with the retrieved b parameter data

24.0.63 read_gswp_ksat (Source File: read_gswp_ksat.F90)

REVISION HISTORY:

12 Sep 2005: James Geiger; Initial Specification

INTERFACE:

```
subroutine read_gswp_ksat(n, array)
```

USES:

```
#if ( defined USE_NETCDF )
    use netcdf
#endif
    use lisdrv_module, only : lis,lisdom
    use lis_logmod, only : lis_log_msg

    implicit none
```

ARGUMENTS:

integer, intent(in)	:: n
real, intent(inout)	:: array(lis%glbnch(n))

DESCRIPTION:

This subroutine retrieves the saturated hydraulic conductivity data in the GSWP data order
The arguments are:

n index of the nest

array output field with the retrieved saturated hydraulic conductivity

24.0.64 read_gswp_psisat (Source File: read_gswp_psisat.F90)

REVISION HISTORY:

12 Sep 2005: James Geiger; Initial Specification

INTERFACE:

```
subroutine read_gswp_psisat(n, array)
```

USES:

```
#if ( defined USE_NETCDF )
    use netcdf
#endif
    use lisdrv_module, only : lis,lisdom
    use lis_logmod, only : lis_log_msg

    implicit none
```

ARGUMENTS:

integer, intent(in)	:: n
real, intent(inout)	:: array(lis%glbnch(n))

DESCRIPTION:

This subroutine retrieves the saturated matric potential data in the GSWP data order
The arguments are:

n index of the nest

array output field with the retrieved saturated matric potential

24.0.65 read_gswp_soilclass (Source File: read_gswp_soilclass.F90)

REVISION HISTORY:

12 Sep 2005: James Geiger; Initial Specification

INTERFACE:

```
subroutine read_gswp_soilclass(n, array)
```

USES:

```
#if ( defined USE_NETCDF )
  use netcdf
#endif
  use lisdrv_module, only : lis, lisdom
  use lis_logmod, only : lis_log_msg

  implicit none
```

ARGUMENTS:

integer, intent(in)	:: n
real, intent(inout)	:: array(lis%lnc(n),lis%lnr(n))

DESCRIPTION:

This subroutine retrieves the soil texture data in the GSWP data order
The arguments are:

n index of the nest

array output field with the retrieved soil texture data

24.0.66 read_gswpclay (Source File: read_gswpclay.F90)

REVISION HISTORY:

15 Jun 2005: James Geiger; Initial Specification

INTERFACE:

```
subroutine read_gswpclay(n, array)
```

USES:

```

#if ( defined USE_NETCDF )
  use netcdf
#endif
  use lisdrv_module, only : lis, lisdom
  use lis_logmod, only : lis_log_msg

  implicit none

```

ARGUMENTS:

integer, intent(in) real, intent(inout)	$:: n$ $:: \text{array}(\text{lis}\%lnc(n), \text{lis}\%lnr(n))$
--	---

DESCRIPTION:

This subroutine retrieves the clay fraction data in the GSWP data order
The arguments are:

n index of the nest

array output field with the retrieved clay fraction data

24.0.67 read_gswpsand (Source File: read_gswpsand.F90)

REVISION HISTORY:

15 Jun 2005: James Geiger; Initial Specification

INTERFACE:

subroutine read_gswpsand(n,array)

USES:

```

#if ( defined USE_NETCDF )
  use netcdf
#endif
  use lisdrv_module, only : lis, lisdom
  use lis_logmod, only : lis_log_msg

  implicit none

```

ARGUMENTS:

integer, intent(in) real, intent(inout)	$:: n$ $:: \text{array}(\text{lis}\%lnc(n), \text{lis}\%lnr(n))$
--	---

DESCRIPTION:

This subroutine retrieves the sand fraction data in the GSWP data order
The arguments are:

n index of the nest

array output field with the retrieved sand fraction data

24.0.68 read_gswpsilt (Source File: read_gswpsilt.F90)

REVISION HISTORY:

15 Jun 2005: James Geiger; Initial Specification

INTERFACE:

```
subroutine read_gswpsilt(n,array)
```

USES:

```
#if ( defined USE_NETCDF )
  use netcdf
#endif
  use lisdrv_module, only : lis, lisdom
  use lis_logmod, only : lis_log_msg

  implicit none
```

ARGUMENTS:

integer, intent(in)	:: n
real, intent(inout)	:: array(lis%lnc(n),lis%lnr(n))

DESCRIPTION:

This subroutine retrieves the silt fraction data in the GSWP data order
The arguments are:

n index of the nest

array output field with the retrieved silt fraction data

24.0.69 read_lambert_faocl (Source File: read_lambert_faocl.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_lambert_faocl(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils

  implicit none
```

ARGUMENTS:

integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))

DESCRIPTION:

This subroutine retrieves FAO clay fraction data and reprojects it to the lambert conformal projection.
The arguments are:

n index of the nest

array output field with the retrieved clay fraction

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in lambert conformal projection

24.0.70 read_lambert_faocolor (Source File: read_lambert_faocolor.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_lambert_faocolor( n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: n
real,       intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO soil color data and reprojects it to the lambert conformal projection.
The arguments are:

n index of the nest

array output field with the retrieved soil color

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in lambert conformal projection

24.0.71 read_lambert_faosand (Source File: read_lambert_faosand.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_lambert_faosand(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real,intent(inout) :: array(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO sand fraction data and reprojects it to the lambert conformal projection.
The arguments are:

n index of the nest

array output field with the retrieved sand fraction

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in lambert conformal projection

24.0.72 read_lambert_faosilt (Source File: read_lambert_faosilt.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_lambert_faosilt(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO silt fraction data and reprojects it to the lambert conformal projection.
The arguments are:

n index of the nest

array output field with the retrieved silt fraction

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in lambert conformal projection

24.0.73 read_lambert_statsgoclay (Source File: read_lambert_statsgoclay.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_lambert_statsgoclay(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO clay fraction data and reprojects it to the lambert conformal projection.
The arguments are:

n index of the nest

array output field with the retrieved clay fraction

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in lambert conformal projection

24.0.74 read_lambert_statsgosand (Source File: read_lambert_statsgosand.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_lambert_statsgosand(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO sand fraction data and reprojects it to the lambert conformal projection. The arguments are:

n index of the nest

array output field with the retrieved sand fraction

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in lambert conformal projection

24.0.75 read_lambert_statsgosilt (Source File: read_lambert_statsgosilt.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_lambert_statsgosilt(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO silt fraction data and reprojects it to the lambert conformal projection.
The arguments are:

n index of the nest

array output field with the retrieved silt fraction

The routines invoked are:

ij_to_latlon (7.1.9)
computes the lat lon values in lambert conformal projection

24.0.76 read_lambert_statsgotexture (Source File: **read_lambert_statsgotexture.F90**)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_lambert_statsgotexture(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer,intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO soil texture data and reprojects it to the lambert conformal projection.
The arguments are:

n index of the nest

array output field with the retrieved soil texture

The routines invoked are:

ij_to_latlon (7.1.9)
computes the lat lon values in lambert conformal projection

24.0.77 read_faocl (Source File: read_faocl.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_faocl(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_fileIOMod, only : lis_open_file
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: n
real,       intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO clay fraction data and reprojects it to the latlon projection.
The arguments are:

n index of the nest

array output field with the retrieved clay fraction

24.0.78 read_faocolor (Source File: read_faocolor.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_faocolor(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_fileIOMod, only : lis_open_file
```

```
implicit none
```

ARGUMENTS:

```
integer,      intent(in)    :: n
real,        intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO soil color data and reprojects it to the latlon projection.
The arguments are:

n index of the nest

array output field with the retrieved soil color

24.0.79 read_faosand (Source File: read_faosand.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_faosand(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n !nest index
real,      intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO sand fraction data and reprojects it to the latlon projection.

The arguments are:

n index of the nest

array output field with the retrieved sand fraction

24.0.80 read_faosilt (Source File: read_faosilt.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_faosilt(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_fileIOMod, only : lis_open_file
```

```
implicit none
```

ARGUMENTS:

```
integer,      intent(in)    :: n
real,       intent(inout)   :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO silt fraction data and reprojects it to the latlon projection.

The arguments are:

n index of the nest

array output field with the retrieved silt fraction

24.0.81 read_statsgoclay (Source File: read_statsgoclay.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_statsgoclay(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_fileIOMod, only : lis_open_file
use lis_logmod, only : logunit
```

```
implicit none
```

ARGUMENTS:

```
integer,      intent(in)    :: n
real,        intent(inout)   :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO clay fraction data and reprojects it to the latlon projection.
The arguments are:

n index of the nest

array output field with the retrieved clay fraction

24.0.82 read_statsgososand (Source File: read_statsgosand.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_statsgosand(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
```

```
implicit none
```

ARGUMENTS:

```
integer,      intent(in)    :: n
real,        intent(inout)   :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO sand fraction data and reprojects it to the latlon projection.
The arguments are:

n index of the nest

array output field with the retrieved sand fraction

24.0.83 read_statsgosilt (Source File: read_statsgosilt.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_statsgosilt(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_fileIOMod, only : lis_open_file
use lis_logmod, only : logunit
```

```
implicit none
```

ARGUMENTS:

```
integer,      intent(in)    :: n
real,        intent(inout)   :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO silt fraction data and reprojects it to the latlon projection.
The arguments are:

n index of the nest

array output field with the retrieved silt fraction

24.0.84 read_statsgotexture (Source File: read_statsgotexture.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_statsgotexture(n,array)
```

USES:

```
use lisdrv_module, only : lis
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
```

```
implicit none
```

ARGUMENTS:

```
integer,intent(in)    :: n
integer,intent(inout)   :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO soil texture data and reprojects it to the latlon projection.
The arguments are:

n index of the nest

array output field with the retrieved soil texture

24.0.85 read_merc_faocl (Source File: read_merc_faocl.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_merc_faocl(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only      : masterproc
use lis_logmod, only   : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO clay fraction data and reprojects it to the mercator projection.
The arguments are:

n index of the nest

array output field with the retrieved clay fraction

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in the mercator projection

24.0.86 read_merc_faocolor (Source File: read_merc_faocolor.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_merc_faocolor(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO soil color data and reprojects it to the mercator projection.
The arguments are:

n index of the nest

array output field with the retrieved soil color

The routines invoked are:

ij_to_latlon (7.1.9)
computes the lat lon values in mercator projection

24.0.87 read_merc_faosand (Source File: read_merc_faosand.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_merc_faosand(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only      : masterproc
use lis_logmod, only   : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO sand fraction data and reprojects it to the mercator projection.
The arguments are:

n index of the nest

array output field with the retrieved sand fraction

The routines invoked are:

ij_to_latlon (7.1.9)
computes the lat lon values in the mercator projection

24.0.88 read_merc_faosilt (Source File: read_merc_faosilt.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_merc_faosilt(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only      : masterproc
use lis_logmod, only   : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO silt fraction data and reprojects it to the mercator projection.
The arguments are:

n index of the nest

array output field with the retrieved silt fraction

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in the mercator projection

24.0.89 read_merc_statsgoclay (Source File: read_merc_statsgoclay.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_merc_statsgoclay(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only      : masterproc
use lis_logmod, only   : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO clay fraction data and reprojects it to the mercator projection.
The arguments are:

n index of the nest

array output field with the retrieved clay fraction

The routines invoked are:

ij_to_latlon (7.1.9)
computes the lat lon values in mercator projection

24.0.90 read_merc_statsgosand (Source File: **read_merc_statsgosand.F90**)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_merc_statsgosand(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO sand fraction data and reprojects it to the mercator projection.
The arguments are:

n index of the nest

array output field with the retrieved sand fraction

The routines invoked are:

ij_to_latlon (7.1.9)
computes the lat lon values in mercator projection

24.0.91 read_merc_statsgosilt (Source File: read_merc_statsgosilt.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_merc_statsgosilt(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real,      intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO silt fraction data and reprojects it to the mercator projection.
The arguments are:

n index of the nest

array output field with the retrieved silt fraction

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in mercator projection

24.0.92 read_merc_statsgotexture (Source File: read_merc_statsgotexture.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_merc_statsgotexture(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer,intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO soil texture data and reprojects it to the mercator projection.
The arguments are:

n index of the nest

array output field with the retrieved soil texture

The routines invoked are:

ij_to_latlon (7.1.9)
computes the lat lon values in mercator projection

24.0.93 read_polar_faocl (Source File: read_polar_faocl.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_polar_faocl(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO clay fraction data and reprojects it to the polar stereographic projection.
The arguments are:

n index of the nest

array output field with the retrieved clay fraction

The routines invoked are:

ij_to_latlon (7.1.9)
computes the lat lon values in polar stereographic projection

24.0.94 read_polar_faocolor (Source File: read_polar_faocolor.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_polar_faocolor(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO soil color data and reprojects it to the polar stereographic projection.
The arguments are:

n index of the nest

array output field with the retrieved soil color

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in polar stereographic projection

24.0.95 read_polar_faosand (Source File: read_polar_faosand.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_polar_faosand(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO sand fraction data and reprojects it to the polar stereographic projection.
The arguments are:

n index of the nest

array output field with the retrieved sand fraction

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in polar stereographic projection

24.0.96 read_polar_faosilt (Source File: read_polar_faosilt.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_polar_faosilt(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves FAO silt fraction data and reprojects it to the polar stereographic projection.
The arguments are:

n index of the nest

array output field with the retrieved silt fraction

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in polar stereographic projection

24.0.97 read_polar_statsgoclay (Source File: read_polar_statsgoclay.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_polar_statsgoclay(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO clay fraction data and reprojects it to the polar stereographic projection. The arguments are:

n index of the nest

array output field with the retrieved clay fraction

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in polar stereographic projection

24.0.98 read_polar_statsgosand (Source File: read_polar_statsgosand.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_polar_statsgosand(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO sand fraction data and reprojects it to the polar stereographic projection.

The arguments are:

n index of the nest

array output field with the retrieved sand fraction

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in polar stereographic projection

24.0.99 read_polar_statsgosilt (Source File: **read_polar_statsgosilt.F90**)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_polar_statsgosilt(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only      : masterproc
use lis_logmod, only    : logunit
use map_utils
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO silt fraction data and reprojects it to the polar stereographic projection.

The arguments are:

n index of the nest

array output field with the retrieved silt fraction

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in polar stereographic projection

24.0.100 read_polar_statsgotexture (Source File: read_polar_statsgotexture.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_polar_statsgotexture(n,array)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer,intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves STATSGO soil texture data and reprojects it to the polar stereographic projection. The arguments are:

n index of the nest

array output field with the retrieved soil texture

The routines invoked are:

ij_to_latlon (7.1.9)
computes the lat lon values in polar stereographic projection

24.0.101 read_afwa_soils (Source File: read_afwa_soils.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_afwa_soils(n,array)
```

USES:

```
use lisdrv_module, only : lis
use lis_fileIOMod, only : lis_open_file
use spmdMod, only : masterproc
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer,intent(in) :: n
integer,intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves the AFWA soil texture data and reprojects it to the latlon projection. The arguments are:

n index of the nest

array output field with the retrieved soil texture

25 Snow

This section describes the routines to ingest various sources of snow datasets in different LIS domains

25.0.102 read_llsnowdepth (Source File: **read_llsnowdepth.F90**)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_llsnowdepth(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This subroutine retrieves the AFWA snowdepth data and reprojects it to the equidistant cylindrical projection. The input data is in inches

25.0.103 read_lcsnowdepth (Source File: **read_lcsnowdepth.F90**)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_lcsnowdepth(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This subroutine retrieves the AFWA snowdepth data and reprojects it to the lambert conformal projection. The input data is in inches

25.0.104 read_mercsnowdepth (Source File: `read_mercsnowdepth.F90`)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_mercsnowdepth(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This subroutine retrieves the AFWA snowdepth data and reprojects it to the mercator projection. The input data is in inches

25.0.105 read_polarsnowdepth (Source File: `read_polarsnowdepth.F90`)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_polarsnowdepth(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This subroutine retrieves the AFWA snowdepth data and reprojects it to the polar stereographic projection. The input data is in inches

26 Topography

This section describes the routines to ingest various sources of topography data in different LIS domains

26.0.106 read_lambert_elev_gtopo30 (Source File: `read_lambert_elev_gtopo30.F90`)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_lambert_elev_gtopo30(n,elev)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_fileIOMod, only : lis_open_file
use spmdMod
use lis_logmod, only :lis_log_msg
use map_utils
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real :: elev(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static elevation data from the GTOPO30 source and reprojects it to the lambert conformal projection.

The arguments are:

n index of the nest

elev output field with the retrieved elevation

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in lambert conformal projection

26.0.107 read_elev_gtopo30 (Source File: `read_elev_gtopo30.F90`)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_elev_gtopo30(n,elev)
```

USES:

```

use lisdrv_module, only : lis
use lis_fileIOMod, only : lis_open_file
use spmdMod
use lis_logmod, only :lis_log_msg

implicit none

```

ARGUMENTS:

```

integer, intent(in) :: n
real                 :: elev(lis%lnc(n), lis%lnr(n))

```

DESCRIPTION:

This subroutine retrieves static elevation data from the GTOPO30 source and reprojects it to the latlon projection.

The arguments are:

n index of the nest

elev output field with the retrieved elevation

26.0.108 read_slope_gtopo30 (Source File: **read_slope_gtopo30.F90**)

REVISION HISTORY:

22 Dec 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_slope_gtopo30(n,slope)
```

USES:

```

use lisdrv_module, only : lis
use lis_fileIOMod, only : lis_open_file
use spmdMod
use lis_logmod, only :lis_log_msg

implicit none

```

ARGUMENTS:

```

integer, intent(in) :: n
real                 :: slope(lis%lnc(n), lis%lnr(n))

```

DESCRIPTION:

This subroutine retrieves static slope data from the GTOPO30 source and reprojects it to the latlon projection.

The arguments are:

n index of the nest

slope output field with the retrieved slope

26.0.109 read_aspect_gtopo30 (Source File: read_aspect_gtopo30.F90)

REVISION HISTORY:

22 Dec 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_aspect_gtopo30(n,aspect)
```

USES:

```
use lisdrv_module, only : lis
use lis_fileIOMod, only : lis_open_file
use spmdMod
use lis_logmod, only :lis_log_msg
```

26.0.110 read_curv_gtopo30 (Source File: read_curv_gtopo30.F90)

REVISION HISTORY:

22 Dec 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_curv_gtopo30(n,curv)
```

USES:

```
use lisdrv_module, only : lis
use lis_fileIOMod, only : lis_open_file
use spmdMod
use lis_logmod, only :lis_log_msg

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real :: curv(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static curvature data from the GTOPO30 source and reprojects it to the latlon projection.

The arguments are:

n index of the nest

curv output field with the retrieved curvature

26.0.111 read_merc_elev_gtopo30 (Source File: read_merc_elev_gtopo30.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_merc_elev_gtopo30(n,elev)
```

USES:

```
use lisdrv_module, only : lis,lisdom  
use lis_fileIOMod, only : lis_open_file  
use spmdMod  
use lis_logmod, only :lis_log_msg  
use map_utils
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n  
real :: elev(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static elevation data from the GTOPO30 source and reprojects it to the mercator projection.

The arguments are:

n index of the nest

elev output field with the retrieved elevation

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in mercator projection

26.0.112 read_polar_elev_gtopo30 (Source File: read_polar_elev_gtopo30.F90)

REVISION HISTORY:

03 Sept 2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_polar_elev_gtopo30(n,elev)
```

USES:

```
use lisdrv_module, only : lis,lisdom  
use lis_fileIOMod, only : lis_open_file  
use spmdMod  
use lis_logmod, only :lis_log_msg  
use map_utils
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real :: elev(lis%lnc(n), lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static elevation data from the GTOPO30 source and reprojects it to the polar stereographic projection.

The arguments are:

n index of the nest

elev output field with the retrieved elevation

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in polar stereographic projection

27 Bottom Temperature

This section describes the routines to ingest various sources of bottom boundary temperature data in different LIS domains

27.0.113 read_gswptbot (Source File: **read_gswptbot.F90**)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_gswptbot(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_logmod, only : lis_log_msg
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static, bottom temperature data in the GSWP data order

The arguments are:

n index of the nest

array output field with the retrieved bottom temperature

27.0.114 read_lambert_statictbot (Source File: read_lambert_statictbot.F90)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_lambert_statictbot(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom  
use lis_fileIOMod, only : lis_open_file  
use map_utils
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n  
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static, bottom temperature data and reprojects it to the lambert conformal projection.

The arguments are:

n index of the nest

array output field with the retrieved bottom temperature

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in lambert conformal projection

27.0.115 read_statictbot (Source File: read_statictbot.F90)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_statictbot(n, array)
```

USES:

```
use lisdrv_module, only : lis  
use lis_fileIOMod, only : lis_open_file, lis_read_file  
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n  
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static, bottom temperature data and reprojects it to the latlon projection.
The arguments are:

n index of the nest
array output field with the retrieved bottom temperature

27.0.116 **read_merc_statictbot** (Source File: **read_merc_statictbot.F90**)

REVISION HISTORY:

25 Jan 2006: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_merc_statictbot(n, array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_fileIOMod, only : lis_open_file
use map_utils
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static, bottom temperature data and reprojects it to the mercator projection.
The arguments are:

n index of the nest
array output field with the retrieved bottom temperature
The routines invoked are:
ij_to_latlon (7.1.9)
computes the lat lon values in mercator projection

27.0.117 **read_polar_statictbot** (Source File: **read_polar_statictbot.F90**)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_polar_statictbot(n,array)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use lis_fileIOMod, only : lis_open_file
use map_utils

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves static, bottom temperature data and reprojects it to the polar stereographic projection.

The arguments are:

n index of the nest

array output field with the retrieved bottom temperature

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in polar stereographic projection

27.0.118 read_AFWAtbot (Source File: read_afwatbot.F90)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_AFWAtbot(n, array)
```

USES:

```
use lisdrv_module, only : lis
use lis_fileIOMod, only : lis_open_file, lis_read_file

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real, intent(inout) :: array(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine retrieves AFWA bottom temperature data and reprojects it to the latlon projection.
The arguments are:

n index of the nest

array output field with the retrieved bottom temperature

Part X

Base forcing analyses in LIS

28 Overview

This section contains various model-based and satellite-based meteorological forcing analyses implemented in LIS. These include the algorithms from the AGRMET system of the Air Force Weather Agency (AFWA), Global Data Assimilation System (GDAS) from NCEP, Goddard Earth Observing System (GEOS) from NASA GSFC, forcing products from the ECMWF, Global Soil Wetness Project (GSWP) and the North American Land Data Assimilation System (NLDAS). The details of each scheme are described in the following sections.

29 AGRMET

The Agricultural Meteorological Modeling System (AGRMET) developed at the United States Air Force Weather Agency (AFWA) is a near real-time operational system that generates land surface conditions for agricultural applications. AGRMET generates the shortwave radiation budget at the surface using the algorithm of [20]. The effects of cloud and snow cover are incorporated using the AFWA RTNEPH 3-hourly global cloud analysis [8]. The satellite images from the polar orbiting satellites of the Defense Meteorological Satellite Program (DMSP) and NOAA satellites systems are used to generate the RTNEPH products. The AGRMET analysis are available on two polar stereographic grids (one for each hemisphere) at 48km resolution. Starting late June 2002, RTNEPH was replaced by a new integrated product called the World Wide Merged Cloud Analysis (WWMCA). In addition to the RTNEPH, the WWMCA product incorporates geostationary satellite images as well, at a higher spatial resolution of 24km.

29.1 Fortran: Module Interface agrmetdomain_module (Source File: agrmetdomain_module.F90)

This module contains the variables and data structures used for the implementation of Air Force Weather Agency (AFWA)'s Agricultural Meteorological Model (AGRMET) analyses to produce meteorological forcing variables for land surface modeling. The algorithms use AFWA's unique database of observational and satellite-based meteorological fields. The input data includes data from Surface Observations database, CDFS II model, and the Special Sensor Microwave/Imager (SSM/I) database. External inputs come from either the NCEP Global Forecast System (GFS), previously known as the Aviation (AVN) model, or the Navy Operational Global Analysis and Prediction System (NOGAPS). AGRMET algorithms also include sophisticated precipitation analysis to compute global precipitation at every grid point around the globe. Estimates of precipitation based on satellite data and climatological tables are blended with surface observations to form a global dataset.

Since the algorithms use a large number of datasets for each analyses, the code assumes the following hierarchy for each instance. For example, if the root directory for storing the files is 'FORCING/AFWA/' and the current instance is decemeber 1st, 2005, the files are stored under 'FORCING/AFWA/20051201/' directory. Under this directory the files should be stored under the following directories.

```
CDMS    : surface and precip obs (sfcobs_* and preobs_*)
GEO     : GEOPRECIP files (prec08* and rank08*)
SFCALC  : processed surface fields (sfcc*)
GFS     : GFS data (MT.avn*)
PRECIP  : processed precip obs (presav_*) and the merged precip
          output (premrgp*)
SSMI    : SSM/I data (ssmira_*)
WWMCA   : WWMCA data (WWMCA*)
```

The implementation of AGRMET algorithms use a common derived data type `agrmet_struct` that includes variables that specify runtime options, information needed for geospatial transformation and other variables. These variables are described below.

`radProcessInterval` interval in hours to perform a radiation analysis

pcpProcessInterval interval in hours to perform a precipitation analysis

readgfsInterval interval in hours to read the GFS data

readsneqvInterval interval in hours to read the snow water equivalent data

radProcessAlarmTime alarm time to keep track of the radiation processing frequency

pcpProcessAlarmTime alarm time to keep track of the precip processing frequency

gfsAlarmTime alarm time to keep track of the GFS data reading frequency

pcpclimoAlarmTime alarm time to keep track of precip climatology reading frequency

readsneqvAlarmTime alarm time to keep track of the snow water equivalent reading frequency

agrmadir root directory for the input files used in different analyses.

climodir precip climatology directory

sneqvfile snow density file

maskfile landmask file used in the AGRMET analyses

terrainfile terrain file used in the AGRMET analyses

sfcntmfile file with the spreading radii used for the barnes analysis on the GFS and surface obs.

analysisdir directory to write the AGRMET intermediate analyses.

agrmetime1 The nearest, previous hourly instance of the incoming data (as a real time).

agrmetime2 The nearest, next hourly instance of the incoming data (as a real time).

agrmepcptime1 The nearest, previous hourly instance of the precip processing (as a real time).

agrmepcptime2 The nearest, next hourly instance of the precip processing (as a real time)

pcpobswch choice for reading precip observations

pwschw choice for using present/past weather estimate

raswch choice for reading SSM/I data

cdfs2swch choice for using CDFS2 estimate

clswch choice for using precip climatology

geoswch choice for using GEOPRECIP data

razero choice for using SSM/I zeros

salp snow distribution shape parameter

clmult alternate monthly weighting factor

mnpr minimum 3 hour climo precip value required to generate a non-zero CDFS2 total cloud based precip estimate

mxpr maximum 3 hour climo precip value required to generate a non-zero CDFS2 total cloud based precip estimate

mnpd minimum precip-per-precip day multiplier used to generate a non-zero CDFS2 total cloud based precip estimate

mxdp maximum precip-per-precip day multiplier used to generate a non-zero CDFS2 total cloud based precip estimate

cldth cloud threshold to generate a CDFS2 based precipitation estimate

mdmv median cloud cover percentage to move to for the CDFS2 based precipitation estimate

mdpe percent to move to median cloud cover percentage to move to for the CDFS2 based precipitation

ovpe overcast percentage to move to for CDFS2 based precipitation estimate

cdfs2int CDFS2 time interval to look for cloud amount

pcap 3 hour maximum precip ceiling

wndwgt weighting factor for first guess winds

minwnd minimum allowable wind speed on the agrmet grid

gridspan value indicating the span of the LIS domain (1-NH only, 2-SH only, 3-span includes both NH and SH)

mo1 number of elements in the NH for the LIS grid

mo2 number of elements in the SH for the LIS grid

hemi_nc number of columns (in the east-west dimension) for each hemisphere

hemi_nr number of rows (in the north-south dimension) for each hemisphere

shemi value indicating the starting index of hemisphere loops (1-NH only, 2-SH only, 1-span includes both NH and SH)

nhemi value indicating the ending index of hemisphere loops (1-NH only, 2-SH only, 2-span includes both NH and SH)

mi number of elements in the input grid

imax x-dimension size of the input grid

jmax y-dimension size of the input grid

sneqv snow water equivalent data

land landmask data

veg vegtype data

alt terrain data

irad radius of observation points for surface OBS

radius radius of observations points for precip OBS

rlat1_nh latitudes of the input grid in the northern hemisphere to be used for bilinear interpolation

rlat1_sh latitudes of the input grid in the southern hemisphere to be used for bilinear interpolation

rlon1_nh longitudes of the input grid in the northern hemisphere to be used for bilinear interpolation

rlon1_sh longitudes of the input grid in the southern hemisphere to be used for bilinear interpolation

n111_nh,n121_nh,n211_nh,n221_nh neighbor information of the input grid in the northern hemisphere to be used for bilinear interpolation

n111_sh,n121_sh,n211_sh,n221_sh neighbor information of the input grid in the southern hemisphere to be used for bilinear interpolation

w111_nh,w121_nh,w211_nh,w221_nh bilinear interpolation weights in the northern hemisphere

w111_sh,w121_sh,w211_sh,w221_sh bilinear interpolation weights in the southern hemisphere

n112_nh, n112_sh neighbor information of the input grid for neighbor interpolation

smask1 landmask to used to fill any gaps in bilinear interpolation due to mismatches in the LIS and AGRMET masks

smask2 landmask to used to fill any gaps in neighbor interpolation due to mismatches in the LIS and AGRMET masks

fillflag1 flag to check for filling gaps due to LIS and AGRMET mask mismatches, for bilinear interpolation

fillflag2 flag to check for filling gaps due to LIS and AGRMET mask mismatches, for neighbor interpolation

cliprc1,cliprc2 month 1 and 2 3-hour precip climo values

clippd1,clippd2 month 1 and 2 precip-per-precip day amount

cirtn1,cirtn2 month 1 and 2 climatological rtneph percent cloud cover

cliprc 3 hour climo precip used for estimate

clippd climatological precip-per-precip day amount

cirtn climatological rtneph percent cloud cover

mrgp merged precip amounts

pcp_start flag indicating the start of the precip analysis

agr_hgt_c geopotential heights on the AGRMET grid for the current time

agr_rh_c relative humidity on the AGRMET grid for the current time

agr_tmp_c temperature on the AGRMET grid for the current time

agr_wspd_c wind speed on the AGRMET grid for the current time

agr_hgt_p geopotential heights on the AGRMET grid from the previous 6 hr

agr_rh_p relative humidity on the AGRMET grid from the previous 6 hr

agr_tmp_p temperature on the AGRMET grid from the previous 6 hr

agr_wspd_p wind speed on the AGRMET grid from the previous 6 hr

step time loop counter used to calculate time interpolation weights

sfcprs first guess pressure fields on the AGRMET grid

srcrlh first guess relative humidity fields on the AGRMET grid

sfctmp first guess temperature fields on the AGRMET grid

lasprs first guess pressure fields for ending 6 hour time on the AGRMET grid

lasrlh first guess relative humidity fields for ending 6 hour time on the AGRMET grid

lasspd first guess wind speed fields for ending 6 hour time on the AGRMET grid

lastmp first guess temperature fields for ending 6 hour time on the AGRMET grid

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: defineNativeAGRMET      !defines the native resolution of
                                         !the input datasets and computes
                                         !interpolation weights
```

PUBLIC TYPES:

```
public :: agrmet_struct
```

29.1.1 defineNativeAGRMET (Source File: agrmetdomain_module.F90)

REVISION HISTORY:

25Jul2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine defineNativeAGRMET
```

USES:

```
use lisdrv_module, only: lis
use listime_mgr, only : setMonthlyAlarm
implicit none
```

DESCRIPTION:

Defines the native resolution of the input forcing for AFWA data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V. The routine first reads the runtime configurable options specific to various AGRMET analyses. The interpolation weights to perform the geospatial transforms to the LIS running domain is computed to be used later. This routine also invokes calls to read all the static data on the AGRMET grid such as landmask, terrain, and the spreading radii used for the barnes analyses. All the time-based alarms to read various AGRMET data is also initialize in this routine.

The routines invoked are:

readagrmetrd (29.1.2)

reads the runtime options specified for AGRMET algorithms

read_agrmetmask (29.1.3)
reads the AGRMET landmask

read_agrmetterrain (29.1.5)
reads the AGRMET terrain

read_pcpcntm (29.1.7)
reads the spreading radii for the precipitation analysis on the AGRMET grid

read_sfcalcncntm (29.1.9)
reads the spreading radii for the surface calculations on the AGRMET grid

bilinear_interp_input (7.0.2)
computes the neighbor and weights for bilinear interpolation

neighbor_interp_input (7.0.6)
computes the neighbor, weights for neighbor interpolation

polarToLatLon (7.0.23)
computes lat lon values for AGRMET grid points

setMonthlyAlarm (5.4.12)
sets the alarms for reading precip climatology

read_pcpclimodata (29.1.11)
reads the precip climatology files

29.1.2 readagrmetcrd (Source File: **readagrmetcrd.F90**)

REVISION HISTORY:

29Jul2005; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine readagrmetcrd()
```

USES:

```
use listime_mgr, only : setHourlyAlarm
use lisdrv_module, only : lis, config_lis
use lis_logmod, only : logunit
use LIS_ConfigMod
use agrmetdomain_module, only : agrmet_struc
```

```
implicit none
```

DESCRIPTION:

This routine reads the options specific to AGRMET algorithms from the LIS configuration file.
The routines invoked are:

LIS_ConfigGetAttribute (5.7.5)
read the specified attribute

LIS_ConfigFindLabel (5.7.8)
find the specified label to read the attribute

setHourlyAlarm (5.4.15)
sets alarms for surface calculation, precipitation processing, and reading GFS and snow water equivalent data.

29.1.3 read_agrmetmask (Source File: read_agrmetmask.F90)

REVISION HISTORY:

1 nov 05 Sujay Kumar, Initial specification

INTERFACE:

```
subroutine read_agrmetmask(n)
```

USES:

```
use lisdrv_module, only : lis
use agrmetdomain_module, only : agrmet_struc
use lis_fileIOMod, only : putget
use lis_logmod, only : logunit, lis_abort

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine reads the AGRMET landmask, in polar stereographic projection at 48km.
The arguments are:

n index of the nest

The routines invoked are:

get_agrmetmask_filename (29.1.4)

 puts together the name of the AGRMET landmask file based on the location of the root directory

putget (5.23.1)

 retrieves the landmask data

lis_abort (5.24.2)

 program aborts in case of error in file open or read

29.1.4 get_agrmetmask_filename (Source File: read_agrmetmask.F90)

INTERFACE:

```
subroutine get_agrmetmask_filename(name, dir,hemi)
```

```
    implicit none
```

ARGUMENTS:

```
    integer, intent(in) :: hemi
    character*100      :: name
    character*50       :: dir
```

DESCRIPTION:

This routines generates the name of the AGRMET landmask file, by appending the root directory and the hemisphere information. The name of the file is expected to be: `jdir/_/point_switches_{hh}`, where hh is the 'nh' or 'sh', for northern and southern hemisphere, respectively.

The arguments are:

hemi index of the hemisphere (1-NH, 2-SH)

dir path to the directory containing the landmask file

name created filename

29.1.5 read_agrmetterrain (Source File: `read_agrmetterrain.F90`)

REVISION HISTORY:

1 nov 05 Sujay Kumar, Initial specification

INTERFACE:

```
subroutine read_agrmetterrain(n)
```

USES:

```
use lisdrv_module, only : lis
use agrmetdomain_module, only : agrmet_struct
use lis_fileIOMod, only : putget
use lis_logmod, only : logunit,lis_abort

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine reads the AGRMET terrain data, in polar stereographic projection at 48km.
The arguments are:

n index of the nest

The routines invoked are:

get_agrmetterrain_filename (29.1.6)

 puts together the name of the AGRMET terrain file based on the location of the root directory

putget (5.23.1)

 retrieves the terrain data

lis_abort (5.24.2)

 program aborts in case of error in file open or read

29.1.6 get_agrmetterrain_filename (Source File: read_agrmetterrain.F90)

INTERFACE:

```
subroutine get_agrmetterrain_filename(name, dir,hemi)
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: hemi
character*100      :: name
character*50       :: dir
```

DESCRIPTION:

This routines generates the name of the AGRMET terrain file, by appending the root directory and the hemisphere information. The name of the file is expected to be: `dir/terrain-hh`, where *hh* is the 'nh' or 'sh', for northern and southern hemisphere, respectively.

The arguments are:

hemi index of the hemisphere (1-NH, 2-SH)
dir path to the directory containing the terrain file
name created filename

29.1.7 read_pcpcntm (Source File: read_pcpcntm.F90)

REVISION HISTORY:

1 nov 05 Sujay Kumar, Initial specification

INTERFACE:

```
subroutine read_pcpcntm(n)
```

USES:

```
use lisdrv_module, only : lis
use agrmetdomain_module, only : agrmet_struct
use lis_fileIOMod, only : putget
use lis_logmod, only : logunit,lis_abort
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine reads the spreading radii used in the barnes analyses for AGRMET precipitation processing. The data is in polar stereographic projection at 48km resolution.
The arguments are:

n index of the nest

The routines invoked are:

get_agrmetpcpcntm_filename (29.1.8)
 puts together the name of the spreading radii file (for precip analysis)

putget (5.23.1)
 retrieves the radii data

lis_abort (5.24.2)
 program aborts in case of error in file open or read

29.1.8 get_agrmetpcpcntm_filename (Source File: **read_pcpcntm.F90**)

INTERFACE:

```
subroutine get_agrmetpcpcntm_filename(name,dir,hemi)
    implicit none
```

ARGUMENTS:

```
integer, intent(in) :: hemi
character*50      :: dir
character*100     :: name
```

DESCRIPTION:

This routines generates the name of the spreading radii file for precip, by appending the root directory and the hemisphere information. The name of the file is expected to be: `|dir|/precip-cntm_|hh|`, where hh is the 'nh' or 'sh', for northern and southern hemisphere, respectively.

The arguments are:

hemi index of the hemisphere (1-NH, 2-SH)

dir path to the directory containing the radii file

name created filename

29.1.9 read_sfcalcncntm (Source File: **read_sfcalcncntm.F90**)

REVISION HISTORY:

1 nov 05 Sujay Kumar, Initial specification

INTERFACE:

```
subroutine read_sfcalcncntm(n)
```

USES:

```
use lisdrv_module, only : lis
use agrmetdomain_module, only : agrmet_struct
use lis_fileIOMod, only : putget
use lis_logmod, only : logunit, lis_abort

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine reads the spreading radii used in the barnes analyses for AGRMET surface calculations. The data is in polar stereographic projection at 48km resolution.

The arguments are:

n index of the nest

The routines invoked are:

get_agrmetsfcntm_filename (29.1.10)

 puts together the name of the spreading radii file (for surface calculations)

putget (5.23.1)

 retrieves the radii data

lis_abort (5.24.2)

 program aborts in case of error in file open or read

29.1.10 **get_agrmetsfcntm_filename** (Source File: **read_sfcalcntm.F90**)

INTERFACE:

```
subroutine get_agrmetsfcntm_filename(name, dir,hemi)
```

```
    implicit none
```

ARGUMENTS:

```
    integer, intent(in) :: hemi
```

```
    character*100      :: name
```

```
    character*50       :: dir
```

DESCRIPTION:

This routines generates the name of the spreading radii file for surface calculations, by appending the root directory and the hemisphere information. The name of the file is expected to be: `dir/sfcalc-cntm_hh`, where hh is the 'nh' or 'sh', for northern and southern hemisphere, respectively.

The arguments are:

hemi index of the hemisphere (1-NH, 2-SH)

dir path to the directory containing the radii file

name created filename

29.1.11 read_pcpclimodata (Source File: read_pcpclimodata.F90)

REVISION HISTORY:

```
04 dec 97 initial version.....capt andrus/dnxm(agromet)
31 mar 99 changed retrieval of data from files to new unix based
           system. increased size of arrays and loops to full
           hemisphere ..... mr moore/dnxm
8 oct 99 ported to ibm sp-2, updated prolog, incorporated
           FORTRAN 90 features.....capt hidalgo/agrmet
21 feb 01 reformatted diagnostic prints. added intent attribute
           to arguments.....mr gayno/dnxm
10 jun 02 updated comments to reflect switch from rtneph to
           cdfs2.....mr gayno/dnxm
1 nov 05 Sujay Kumar, Adopted in LIS
```

INTERFACE:

```
subroutine read_pcpclimodata(n)
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr, only : isMonthlyAlarmRinging, computeMonthlyWeights
use agrmetdomain_module, only :agrmet_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: n
```

DESCRIPTION:

This routine retrieves monthly climo data from file and interpolate to the current day.
method:

- retrieve climo for current month.
- if date is 15th, current months climo is valid, no more work needs to be done.
- else
- determine if need to interpolate to previous month,
or to the next month.
- read appropriate month from file and interpolate
values to the current day.

The arguments and variables are:

n index of the nest

days number of days in each month (for leap year)

hemi hemisphere (1=nh, 2=sh)

newday number of days to interpolate to

totday total days in the current, or previous

i,j loop indices

quad9r constant = 9999.0

remarks:

NOTE: although we use cdfs2 cloud analysis data, we use rtnepl climo information in the cdfs2 precip estimate. this is because we do not have a climo database based on cdfs2 as of Jun, 2002. ideally, once we develop a large cdfs2 database, we should create a new climo.

The routines invoked are:

isMonthlyAlarmRinging (5.4.1)

check to see if it time to read a new climo data

getclimortnfilename (29.1.12)

filename for the climatological rtnepl percent cloud cover

getclimoprcfilename (29.1.13)

filename for the precip used for estimate

getclimoppdfilename (29.1.14)

filename for the precip-per-precip day amount

getcli (29.1.15)

retrieves the climo data for any of the above datasets

29.1.12 getclimortnfilename (Source File: `read_pcpclimodata.F90`)

INTERFACE:

```
subroutine getclimortnfilename(name, dir,hemi,mo)
  implicit none
```

ARGUMENTS:

```
integer, intent(in) :: hemi
integer, intent(in) :: mo
character*100      :: name
character*50        :: dir
```

DESCRIPTION:

This routine generates the name of the climatological rtnepl percent cloud cover file, by appending the root directory and the hemisphere information. The name of the file is expected to be: `jdir/jclim_rtn_jmon_jhh`, where mon is the 3-character name of the month and hh is 'nh' or 'sh', for northern and southern hemisphere, respectively.

The arguments are:

hemi index of the hemisphere (1-NH, 2-SH)

mo integer value of month (1-12)

dir path to the directory containing the climo file

name created filename

29.1.13 getclimoprcfilename (Source File: read_pcpclimodata.F90)

INTERFACE:

```
subroutine getclimoprcfilename(name, dir,hemi,mo)  
    implicit none
```

ARGUMENTS:

```
integer, intent(in) :: hemi  
integer, intent(in) :: mo  
character*100      :: name  
character*50       :: dir
```

DESCRIPTION:

This routines generates the name of the 3-hour climo precip file, by appending the root directory and the hemisphere information. The name of the file is expected to be: $\text{dir}_c/\text{clim_prc_}\text{mon}_c\text{_}\text{hh}_c$, where mon is the 3-character name of the month and hh is 'nh' or 'sh', for northern and southern hemisphere, respectively. The arguments are:

hemi index of the hemisphere (1-NH, 2-SH)
mo integer value of month (1-12)
dir path to the directory containing the climo file
name created filename

29.1.14 getclimoppdfilename (Source File: read_pcpclimodata.F90)

INTERFACE:

```
subroutine getclimoppdfilename(name, dir,hemi,mo)  
    implicit none
```

ARGUMENTS:

```
integer, intent(in) :: hemi  
integer, intent(in) :: mo  
character*100      :: name  
character*50       :: dir
```

DESCRIPTION:

This routines generates the name of the precip-per-precip day amount file, by appending the root directory and the hemisphere information. The name of the file is expected to be: $\text{dir}_c/\text{clim_ppd_}\text{mon}_c\text{_}\text{hh}_c$, where mon is the 3-character name of the month and hh is 'nh' or 'sh', for northern and southern hemisphere, respectively.

The arguments are:

hemi index of the hemisphere (1-NH, 2-SH)

mo integer value of month (1-12)

dir path to the directory containing the climo file

name created filename

29.1.15 getcli (Source File: getcli.F90)

REVISION HISTORY:

```
04 dec 97 initial version.....capt andrus/dnxm(agromet)
31 mar 99 changed retrieval of data from files to new unix based
          system. increased size of arrays and loops to full
          hemisphere ..... mr moore/dnxm
 8 oct 99 ported to ibm sp-2, updated prolog, incorporated
          FORTRAN 90 features.....capt hidalgo/agrmet
21 feb 01 reformatted diagnostic prints. added intent attribute
          to arguments.....mr gayno/dnxm
10 jun 02 updated comments to reflect switch from rtneph to
          cdfs2.....mr gayno/dnxm
 1 nov 05 Sujay Kumar, Adopted in LIS
```

INTERFACE:

```
subroutine getcli(n, filename,rtn,clidat)
```

USES:

```
use lisdrv_module, only      : lis
use lis_fileIOMod, only      : putget
use agrmetdomain_module, only : agrmet_struc
use lis_logmod, only         : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: n
character*100            :: filename
real,     intent(out)    :: clidat(agrmet_struc(n)%imax,agrmet_struc(n)%jmax)
integer                  :: rtn
```

DESCRIPTION:

This routine retrieves monthly climo data from file. The same routine is called to retrieve the three types of
precip climo data. climatological rtneph percent cloud cover
3 hour precip climo
precip-per-precip day amount

The arguments and variables are:

n index of the nest

filename name of the climo file

rtn flag to indicate that the data should be read as integers

clidat output climatological values
exists a logical that indicates whether or not a file exists
tempdata a temp array used to read in climo files that are stored as integers

29.1.16 getagrmet (Source File: getagrmet.F90)

REVISION HISTORY:

29 Jul 2005; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine getagrmet(n)
```

USES:

```
use lisdrv_module, only      : lis
use baseforcing_module, only : lisforc
use agrmetdomain_module, only: agrmet_struc
use listime_mngr, only       : get_nstep, isHourlyAlarmRinging, &
                               computeTimeBookEnds,time2date,tick
use lis_logmod, only         : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This is the entry point for calling various AGRMET analyses. At the beginning of a simulation, the code invokes call to read the most recent past data (nearest hourly interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. The strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day. This entry point code also calls different routines based on the running mode. The AGRMET mode will generate the surface analysis, merged precipitaiton, whereas the analysis mode will simply use the previously generated data. . The arguments are:

n index of the nest

The routines invoked are:

isHourlyAlarmRinging (5.4.16)

check to see if the hourly intervals have reached to start reading different data.

computeTimeBookEnds (5.4.17)

computes the previous and past times for the specified input data frequency

readagrmetforcing (29.1.18)

routines that perform the surface calculations, radiation processing (all met forcing except precip)

readagrmetpcpforcing (29.1.60)

routines that perform the precip analysis

readagrmetforcinganalysis (29.1.24)
routine to read analysis of surface fields.

readagrmetpcpforcinganalysis (29.1.63)
routines that reads the merged precip analysis

time2date (5.4.21)
converts time to a date format

tick (5.4.22)
computes the AGRMET read/processing times.

29.1.17 retrieve_agrmetvar (Source File: retrieve_agrmetvar.F90)

REVISION HISTORY:

29Jul2005; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine retrieve_agrmetvar(name,nc,nr,varfield)
```

USES:

```
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

```
character*100      :: name
integer           :: nc,nr
real, intent(inout) :: varfield(nc,nr)
```

DESCRIPTION:

Routine to retrieve variables in the polar stereographic projection, in a hemisphere
The arguments are:

name name of the file to be read from

nc number of columns (east-west dimension) of the data

nr number of rows (north-south dimension) of the data

varfield the retrieved data

29.1.18 readagrmetforcing (Source File: readagrmetforcing.F90)

REVISION HISTORY:

29Jul2005; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine readagrmetforcing(n,order)
```

USES:

```
use lisdrv_module, only      : lis,lisdom
use agrmetdomain_module, only : agrmet_struc
use baseforcing_module, only  : lisforc
use listime_mngr,only        : get_julhr,tick,time2date
use spmdMod, only            : masterproc
use albedo_module, only      : lis_alb
use gfrac_module, only       : lis_gfrac
use snow_module, only        : snow_struc
use lis_logmod, only         : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: order
```

DESCRIPTION:

This routine generates the meteorological forcing variables for the current time. The surface observations along with GFS data is processed to generate estimates of temperature, pressure, wind and relative humidity. The WWMCA cloud data is read in to generate the downward shortwave and longwave radiation fields. This routine also performs the spatial interpolation to the LIS grid and filling any gaps introduced due to mismatches in the LIS and AGRMET masks.

The indices of glbdata1 and glbdata2 correspond to the following variables:

```
1 - 2m air temp
2 - 2m relative humidity
3 - u wind
4 - v wind
5 - surface pressure
6 - longwave
7 - shortwave
8 - precip
```

The arguments and variables are:

order flag indicating which data to be read (order=1, read the previous hourly instance, order=2, read the next hourly instance)

n index of the nest

filename generated filename

hemi index of hemisphere loops

c,r,i,t looping and indexing variables

istat io error stat variable

file_exists check for archived file

varfield interpolated variable

tair interpolated surface temperature

psurf interpolated surface pressure

rlh interpolated relative humidity
vapor_pres vapor pressure value
ip choice of interpolation algorithm
julhr julian hour value
amounts retrieved CDFS2 cloud amounts
types retrieved CDFS2 cloud types
tops retrieved CDFS2 cloud tops
times retrieved CDFS2 pixel times
cldamt_nh, cldamt_sh retrieved cloud amounts for each hemisphere
cldtyp_nh, cldtyp_sh retrieved cloud types for each hemisphere
fog_nh,fog_sh fog present flags for each hemisphere
thres cloud top thresholds
q2sat saturation mixing ratio at the 1st model level
esat saturation vapor pressure
udef undefined variable used for spatial interpolation
doy1,yr1,mo1,da1,hr1,mn1,ss1,try,ts1,backtime1,gmt1 time/date specific variables
fog fog present flag
bare flag to eliminate greenness at a point
albedo snow free albedo
snup snow depth thresholds

The routines invoked are:

agrmet_sfcalc (29.1.29)
 calls the routines to produce an analysis of temperature, pressure winds and relative humidity

find_agrsfc_dataindex (29.1.23)
 computes the data index for the current hour

interp_agrmetvar (29.1.45)
 spatial interpolation of an AGRMET variable to LIS grid

agrmet_fillgaps (29.1.46)
 fills the gaps in the interpolated field due to mismatches in LIS and AGRMET masks

tick (5.4.22)
 determines the input times of different files

time2date (5.4.21)
 converts the time to a date format

agrmet_cdfs_type_filename (29.1.20)
 generates the filename to read CDFS2 cloud types

agrmet_cdfs_pcts_filename (29.1.19)
generates the filename to read CDFS2 cloud amounts

agrmet_cdfs_hgts_filename (29.1.21)
generates the filename to read CDFS2 cloud tops

agrmet_cdfs_pixltime_filename (29.1.22)
generates the filename to read CDFS2 pixel times

get_julhr (5.4.9)
converts the date to a julian hour

loadcloud (29.1.58)
converts CDFS2 data for use in radiation routines

agrmet_svp (29.1.47)
computes saturation vapor pressure and saturation specific humidity

agrmet_longwv (29.1.48)
computes the downward longwave radiation

agrmet_calc_albedo (29.1.49)
computes the snow free albedo for current day

agrmet_tr_coeffs (29.1.50)
computes the transmissivity and reflectivity coefficients

agrmet_solar (29.1.51)
computes the downward shortwave radiation value

29.1.19 agrmet_cdfs_pcts_filename (Source File: **readagrmetforcing.F90**)

INTERFACE:

```
subroutine agrmet_cdfs_pcts_filename(name,dir,hemi,yr,mo,da,hr)
```

```
    implicit none
```

ARGUMENTS:

```
    integer, intent(in) :: hemi
    integer, intent(in) :: yr,mo,da,hr
    character*100      :: name
    character*50        :: dir
```

DESCRIPTION:

This routines generates the name of the CDFS2 cloud amounts file by appending the hemisphere and timestamps to the root directory.

The arguments are:

hemi index of the hemisphere (1-NH, 2-SH)

dir full path to the directory containing the data

name created filename

yr 4 digit year
mo integer value of month (1-12)
da day of the month
hr hour of the day

29.1.20 agrmet_cdfs_type_filename (Source File: readagrmetforcing.F90)

INTERFACE:

```
subroutine agrmet_cdfs_type_filename(name,dir,hemi,yr,mo,da,hr)
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: hemi
integer, intent(in) :: yr,mo,da,hr
character*100      :: name
character*50       :: dir
```

DESCRIPTION:

This routines generates the name of the CDFS2 cloud types file the hemisphere and timestamps to the root directory.

The arguments are:

hemi index of the hemisphere (1-NH, 2-SH)
dir full path to the directory containing the data
name created filename
yr 4 digit year
mo integer value of month (1-12)
da day of the month
hr hour of the day

29.1.21 agrmet_cdfs_hgts_filename (Source File: readagrmetforcing.F90)

INTERFACE:

```
subroutine agrmet_cdfs_hgts_filename(name,dir,hemi,yr,mo,da,hr)
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: hemi
integer, intent(in) :: yr,mo,da,hr
character*100      :: name
character*50       :: dir
```

DESCRIPTION:

This routines generates the name of the CDFS2 cloud tops file with hemisphere and timestamps to the root directory.

The arguments are:

hemi index of the hemisphere (1-NH, 2-SH)
dir full path to the directory containing the data
name created filename
yr 4 digit year
mo integer value of month (1-12)
da day of the month
hr hour of the day

29.1.22 agrmet_cdfs_pixltime_filename (Source File: readagrmeforcing.F90)

INTERFACE:

```
subroutine agrmet_cdfs_pixltime_filename(name,dir,hemi,yr,mo,da,hr)
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: hemi
integer, intent(in) :: yr,mo,da,hr
character*100      :: name
character*50        :: dir
```

DESCRIPTION:

This routines generates the name of the CDFS2 pixel times file. the hemisphere and timestamps to the root directory.

The arguments are:

hemi index of the hemisphere (1-NH, 2-SH)
dir full path to the directory containing the data
name created filename
yr 4 digit year
mo integer value of month (1-12)
da day of the month
hr hour of the day

29.1.23 find_agrsfc_dataindex (Source File: readagrmetforcing.F90)

INTERFACE:

```
subroutine find_agrsfc_dataindex(hr,dindex)
    implicit none
```

ARGUMENTS:

```
integer, intent(in) :: hr
integer, intent(inout) :: dindex
```

DESCRIPTION:

This routine finds index to be used in the SFCALC data for the current instance. The SFCALC processing is done every 6 hours, which produces products for the upcoming hourly intervals. For every hour within a 6 hour window, to retrieve the data, the program will simply use the index returned by this routine, instead of redoing the processing.

The arguments are:

hr the current hour

dindex output data index

29.1.24 readagrmetforcinganalysis (Source File: readagrmetforcinganalysis.F90)

REVISION HISTORY:

29Jul2005; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine readagrmetforcinganalysis(n,order)
```

USES:

```
use lisdrv_module, only      : lis,lisdom
use agrmetdomain_module, only : agrmet_struc
use baseforcing_module, only  : lisforc
use listime_mngr,only        : get_julhr,tick,time2date,tick
use spmdMod, only            : masterproc
use albedo_module, only      : lis_alb
use gfrac_module, only       : lis_gfrac
use snow_module, only        : snow_struc
use lis_logmod, only         : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: order
```

DESCRIPTION:

This routine reads the previously generated surface fields of temperature, relative humidity, wind, and pressures. The WWMCA cloud data is read in to generate the downward shortwave and longwave radiation fields. This routine also performs the spatial interpolation to the LIS grid and filling any gaps introduced due to mismatches in the LIS and AGRMET masks.

The indices of glbdata1 and glbdata2 correspond to the following variables:

```
1 - 2m air temp
2 - 2m relative humidity
3 - u wind
4 - v wind
5 - surface pressure
6 - longwave
7 - shortwave
8 - precip
```

The arguments and variables are:

order flag indicating which data to be read (order=1, read the previous hourly instance, order=2, read the next hourly instance)

n index of the nest

filename generated filename

hemi index of hemisphere loops

c,r,i,t looping and indexing variables

istat io error stat variable

file_exists check for archived file

varfield interpolated variable

tair interpolated surface temperature

psurf interpolated surface pressure

rlh interpolated relative humidity

vapor_pres vapor pressure value

ip choice of interpoation algorithm

julhr julian hour value

amounts retrieved CDFS2 cloud amounts

types retrieved CDFS2 cloud types

tops retrieved CDFS2 cloud tops

times retrieved CDFS2 pixel times

cldamt_nh, cldamt_sh retrieved cloud amounts for each hemisphere

cldtyp_nh, cldtyp_sh retrieved cloud types for each hemisphere

fog_nh,fog_sh fog present flags for each hemisphere

thres cloud top thresholds
q2sat saturation mixing rati at the 1st model level
esat saturation vapor pressure
udef undefined variable used for spatial interpolation
doy1,yr1,mo1,da1,hr1,mn1,ss1,try,ts1,backtime1,gmt1 time/date specific variables
fog fog present flag
bare flag to eliminate greenness at a point
albedo snow free albedo
snup snow depth thresholds

The routines invoked are:

- interp_agrmetvar** (29.1.45)
spatial interpolation of an AGRMET variable to LIS grid
- agrmet_fillgaps** (29.1.46)
fills the gaps in the interpolated field due to mismatches in LIS and AGRMET masks
- tick** (5.4.22)
determines the input times of different files
- time2date** (5.4.21)
converts the time to a date format
- agrmet_cdfs_type_filename** (29.1.20)
generates the filename to read CDFS2 cloud types
- agrmet_cdfs_pcts_filename** (29.1.19)
generates the filename to read CDFS2 cloud amounts
- agrmet_cdfs_hgts_filename** (29.1.21)
generates the filename to read CDFS2 cloud tops
- retrieve_agrmetvar** (29.1.17)
retrieves the specified variable in PS grid
- agrmet_cdfs_pixltime_filename** (29.1.22)
generates the filename to read CDFS2 pixel times
- agrmet_sfctmp_filename** (29.1.25)
generates the name of the surface temperature analysis
- agrmet_sfcrlh_filename** (29.1.26)
generates the name of the relative humidity analysis
- agrmet_sfcpd_filename** (29.1.28)
generates the name of the surface wind analysis
- agrmet_sfcprs_filename** (29.1.27)
generates the name of the surface pressure analysis
- get_julhr** (5.4.9)
converts the date to a julian hour

loadcloud (29.1.58)
converts CDFS2 data for use in radiation routines

agrmet_svp (29.1.47)
computes saturation vapor pressure and saturation specific humidity

agrmet_longwv (29.1.48)
computes the downward longwave radiation

agrmet_tr_coeffs (29.1.50)
computes the transmissivity and reflectivity coefficients

agrmet_calc_albedo (29.1.49)
computes the snow free albedo for current day

agrmet_solar (29.1.51)
computes the downward shortwave radiation value

29.1.25 agrmet_sfctmp_filename (Source File: readagrmetforcinganalysis.F90)

INTERFACE:

```
subroutine agrmet_sfctmp_filename(name,dir,hemi,yr,mo,da,hr)
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: hemi
integer, intent(in) :: yr,mo,da,hr
character*100      :: name
character*50       :: dir
```

DESCRIPTION:

This routines generates the name of the surface temperature file by appending the hemisphere and timestamps to the root directory.

The arguments are:

hemi index of the hemisphere (1-NH, 2-sh)
dir full path to the directory containing the data
name created filename
yr 4 digit year
mo integer value of month (1-12)
da day of the month
hr hour of the day

29.1.26 agrmet_sfcrlh_filename (Source File: readagrmetforcinganalysis.F90)

INTERFACE:

```
subroutine agrmet_sfcrlh_filename(name,dir,hemi,yr,mo,da,hr)
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: hemi
integer, intent(in) :: yr,mo,da,hr
character*100      :: name
character*50       :: dir
```

DESCRIPTION:

This routines generates the name of the relative humidity file by appending the hemisphere and timestamps to the root directory.

The arguments are:

hemi index of the hemisphere (1-nh, 2-sh)
dir full path to the directory containing the data
name created filename
yr 4 digit year
mo integer value of month (1-12)
da day of the month
hr hour of the day

29.1.27 agrmet_sfcpres_filename (Source File: readagrmetforcinganalysis.F90)

INTERFACE:

```
subroutine agrmet_sfcpres_filename(name,dir,hemi,yr,mo,da,hr)
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: hemi
integer, intent(in) :: yr,mo,da,hr
character*100      :: name
character*50       :: dir
```

DESCRIPTION:

This routines generates the name of the surface pressure file by appending the hemisphere and timestamps to the root directory.

The arguments are:

hemi index of the hemisphere (1-nh, 2-sh)
dir full path to the directory containing the data
name created filename
yr 4 digit year
mo integer value of month (1-12)
da day of the month
hr hour of the day

29.1.28 agrmet_sfcspd_filename (Source File: readagrmeforcinganalysis.F90)

INTERFACE:

```
subroutine agrmet_sfcspd_filename(name,dir,hemi,yr,mo,da,hr)
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: hemi
integer, intent(in) :: yr,mo,da,hr
character*100      :: name
character*50       :: dir
```

DESCRIPTION:

This routines generates the name of the surface wind file by appending the hemisphere and timestamps to the root directory.

The arguments are:

hemi index of the hemisphere (1-nh, 2-sh)
dir full path to the directory containing the data
name created filename
yr 4 digit year
mo integer value of month (1-12)
da day of the month
hr hour of the day

29.1.29 agrmet_sfcalc (Source File: agrmet_sfcalc.F90)

INTERFACE:

```
subroutine agrmet_sfcalc(n, sfcpes, sfcrlh, sfcspe, sfctmp,&
    lasprs, lasrlh, lasspe, lastmp)
```

USES:

```
use lisdrv_module, only      : lis
use agrmetdomain_module, only : agrmet_struc
use listime_mgr, only        : isHourlyAlarmRinging, get_julhr,&
    julhr_date
use lis_logmod, only         : logunit
use lis_fileIOMod, only      : putget
use spmdMod
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
real :: sfcpes(2,6,agrmet_struc(n)%imax,agrmet_struc(n)%jmax)
real :: sfcrlh(2,6,agrmet_struc(n)%imax,agrmet_struc(n)%jmax)
real :: sfcspe(2,6,agrmet_struc(n)%imax,agrmet_struc(n)%jmax)
real :: sfctmp(2,6,agrmet_struc(n)%imax,agrmet_struc(n)%jmax)

real :: lasprs(2,agrmet_struc(n)%imax,agrmet_struc(n)%jmax)
real :: lasrlh(2,agrmet_struc(n)%imax,agrmet_struc(n)%jmax)
real :: lasspe(2,agrmet_struc(n)%imax,agrmet_struc(n)%jmax)
real :: lastmp(2,agrmet_struc(n)%imax,agrmet_struc(n)%jmax)
```

DESCRIPTION:

This routine generates the surface fields of temperature, pressure, winds and relative humidity. It reads the GFS data, interpolates first guess height, temperature, moisture, and wind data to the AGRMET grid. These fields are vertically interpolated to each point's elevation. Finally the routine blends the surface observations of temperature, relative humidity and surface wind speed with the first guess fields using a barnes analysis. The arguments and variables are:

n index of the nest

sfcpes first guess pressure fields on the AGRMET grid interpolated to the current time

sfcrlh first guess rh fields on the AGRMET grid interpolated to the current time

sfcspe first guess wind speed fields on the AGRMET grid interpolated to the current time

sfctmp first guess temperature fields on the AGRMET grid interpolated to the current time

lasprs first guess pressure fields for ending 6 hour time

lasrlh first guess rh fields for ending 6 hour time

lasspe first guess wind speed fields for ending 6 hour time

lastmp first guess temperature fields for ending 6 hour time

isize array size for observations

yr1,mo1,da1,hr1,mn1,ss1,yr,mo,da,hr,mn,ss time/date variables
kprs number of isobaric levels for first guess data
prslvls isobaric levels for first guess data
alert_number number of alerts that occur in the program
timetoReadGFS flag to check for GFS reading frequency
radrlh radius of influence- for each AGRMET grid point- of wind speed observations in barnes analysis
radspd radius of influence- for each AGRMET grid point- of wind speed observations in barnes analysis
radtmp radius of influence- for each AGRMET grid point- of temperature observations in barnes analysis
lokrlh standard radii of influence of relative humidity observations in barnes analysis
lokspd standard radii of influence of wind speed observations in barnes analysis
loktmp standard radii of influence of temperature observations in barnes analysis
ri i coordinate of surface observations on the AGRMET grid
rj j coordinate of surface observations on the AGRMET grid
obsrlh array of rlh observations
obsspd array of wind speed observations
obstmp array of temperature observations
obscnt number of retrieved surface observations
jul,julhr,cur_jul,prev_jul julian hour variables
i,j loop indices
order flag to indicate which data is to be read
The routines invoked are:
find_agrfld_starttime (29.1.30)
computes the start time for the FLDBLD processing.
fldbld (29.1.31)
generates the surface fields
get_julhr (5.4.9)
converts the date to a julian hour
fgfill (29.1.40)
determine a first guess field for the current time
getsfc (29.1.42)
retrieve surface observations of temperature, relative humidity, and wind speed from CDMS
sfcalc_barnes (29.1.44)
barnes optimal interpolation

29.1.30 find_agrfld_starttime (Source File: agrmet_sfcalc.F90)

INTERFACE:

```
subroutine find_agrfld_starttime(yr,mo,da,hr,julbeg)
```

USES:

```
use listime_mgr, only : tick, get_julhr  
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: yr  
integer, intent(in) :: mo  
integer, intent(in) :: da  
integer, intent(in) :: hr  
integer, intent(inout) :: julbeg
```

DESCRIPTION:

This routine finds the julian hour to start the AGRMET precip processing from, based on the current input time.

The arguments are:

yr the current year

mo the current month

da the current day

hr the current hour

julbeg output starting julian hour

The routines invoked are:

tick (5.4.22)
computes previous 6 hour time.

get_julhr (5.4.9)
converts the date to a julian hour

29.1.31 fldbld (Source File: fldbld.F90)

REVISION HISTORY:

29 Jul 2005; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine fldbld(n,order,julhr)
```

USES:

```

use lisdrv_module, only : lis
use agrmetdomain_module, only : agrmet_struct
use lis_logmod, only      : logunit, lis_abort
use listime_mgr, only      : julhr_date

implicit none

```

ARGUMENTS:

```

integer, intent(in) :: n
integer, intent(in) :: order

```

DESCRIPTION:

This routine interpolates the first guess height, temperature, moisture, and wind data to the AGRMET grid. The arguments and variables are:

n index of the nest

avnfile name of the first guess file

nunit unix file descriptor

ksec2 array of section 2 information from a grib record see comments for more information.

message error message

iginfo array of grid information from a grib record

ginfo array of grid information from a grib record

gridres the resolution, in degrees, of the first guess grid

alert_number number of alerts that occur in the program

ifguess east/west dimension of first guess grid

jfguess north/south dimension of first guess grid

kprs number of isobaric levels for first guess data

prslvls isobaric levels for first guess data

istat io error stat variable

nmlfil name of first guess namelist file

ksec1 array of grib section 1 information see comments for more specific information

avnflag flag used in grid calculations to account for the fact that point (1,1) on the avn/nogaps grid is at the north/south pole.

center meteorological center that produced the first guess data (7-NCEP, 58-FNMOC)

ierr error code

The routines invoked are:

julhr_date (5.4.23)

converts the julian hour to a date format

getAVNfilename (29.1.32)

generates the first guess AVN filename

getDataLevelsFilename (29.1.34)
generates the first guess namelist filename

fldbld_setup (29.1.35)
read grid specific information from the first guess grib data

fldbld_read (29.1.36)
read AVN or NOGAPS data in grib format

fg_to_agr (29.1.39)
interpolate first guess data to the AGRMET grid

lis_abort (5.24.2)
abort in case of error

29.1.32 getAVNfilename (Source File: `fldbld.F90`)

29.1.33 (Source File: `fldbld.F90`)

subroutine `getAVNfilename(filename, dir, yr,mo,da,hr)`
implicit none *ARGUMENTS:*

```
character(len=*)      :: filename
character(len=*)      :: dir
integer, intent(in)  :: yr,mo,da,hr
```

DESCRIPTION:

This routines generates the name of the timestamped AVN file
The arguments are:

dir full path to the directory containing the data

filename created filename

yr 4 digit year

mo integer value of month (1-12)

da day of the month

hr hour of the day

29.1.34 getDataLevelsFilename (Source File: `fldbld.F90`)

INTERFACE:

```
subroutine getDataLevelsFilename(filename, dir, hr)
  implicit none
```

ARGUMENTS:

```

character(len=*) :: filename
character(len=*) :: dir
integer, intent(in) :: hr

```

DESCRIPTION:

This routines generates the name of the timestamped AVN file
The arguments are:

dir full path to the directory containing the data
filename created filename
hr hour of the day

29.1.35 fldbld_setup (Source File: fldbld_setup.F90)

REVISION HISTORY:

```

03 mar 99 initial version .....capt andrus/dnxm
12 aug 99 ported to IBM SP-2. added intent attributes to
arguments. made grid size dependent variables
dynamically allocatable. made calculation of
jlat and ilon flexible for any lat/lon grid
resolution.....mr gayno/dnxm
12 aug 02 changed variables ilon and jlat to be real instead of
nearest integer.....mr gayno/dnxm

```

20 aug 05 Sujay Kumar, Initial Specification in LIS

INTERFACE:

```

subroutine fldbld_setup( jlat, ilon, imax, jmax, &
avnflag, gridres )
implicit none

```

ARGUMENTS:

integer, intent(in)	:: avnflag
integer, intent(in)	:: imax
integer, intent(in)	:: jmax
real, intent(in)	:: gridres
real, intent(out)	:: ilon (imax, jmax, 2)
real, intent(out)	:: jlat (imax, jmax, 2)

DESCRIPTION:

to calculate the location of the agrmet grid points with respect to the first guess grid.

Method

- loop over the agrmet grid.
- call utility routine pstoll to find latitude and longitude of the agrmet grid point.
- determine the (i,j) coordinate of the agrmet grid point on the first guess grid.

The arguments are:

avnflag flag used in grid calculations to account for the fact that point (1,1) on the avn/nogaps grid is at the north/south pole.

gridres grid resolution, in degrees, of the first guess grid

jlat north/south coordinate of the agrmet grid with respect to the first guess grid

ilon east/west coordinate of the agrmet grid with respect to the first guess grid

imax east/west coordinate of the agrmet grid

jmax north/south dimension of the agrmet grid

29.1.36 fldbld_read (Source File: fldbld_read.F90)

REVISION HISTORY:

```
05 aug 99 initial version.....mr gayno/dnxm
31 mar 00 changed variable "fg_filename" to character*120 to
           be compatable with routine copen.c.....mr gayno/dnxm
07 jan 02 modified for new avn file naming convention.....
           .....mr gayno/dnxm
01 aug 02 removed hard-wired references to isobaric levels of
           first guess source.....mr gayno/dnxm
20 aug 05 Sujay Kumar, Initial Specification in LIS
```

INTERFACE:

```
subroutine fldbld_read( fg_filename, ifguess, jfguess,&
                       wndwgt, minwnd, fg_hgt, fg_rh,&
                       fg_tmp, fg_wspd,&
                       kprs, prslvls, alert_number )
```

USES:

```
use lis_logmod, only : logunit, lis_abort, lis_alert
implicit none
```

ARGUMENTS:

```
character(len=*), intent(in) :: fg_filename
integer,         intent(in)  :: ifguess
integer,         intent(in)  :: jfguess
real,            intent(in)  :: minwnd
real,            intent(in)  :: wndwgt
integer,         intent(in)  :: kprs
real,            intent(out) :: fg_hgt      ( ifguess,jfguess,kprs )
real,            intent(out) :: fg_rh       ( ifguess,jfguess,kprs )
real,            intent(out) :: fg_tmp      ( ifguess,jfguess,kprs )
real,            intent(out) :: fg_wspd     ( ifguess,jfguess )
integer,         intent(in)  :: prslvls    ( 30 )
integer,         intent(inout) :: alert_number
```

DESCRIPTION:

to read avn or nogaps data in grib format.

Method

- open file and allocate variables.
- read in each grib record.
- retrieve section 1 information.
- if data is what we need, store it in the proper arrays.
also, keep track of what has been read in using
counter variables.
- check forecast hour of data, if it is not the analysis
or zero hour forecast, then send an alert message
to warn of possible degradation.
- check counter variables. if data is missing abort.
- if nogaps, convert dewpoint depression to relative
humidity.
- calculate surface wind speed from u and v components.
multiply by wind weighting factor. restrict lowest
wind speed to value of minwnd.

fg_filename name, including path, of the first guess file being read in

ifguess east-west dimension of first guess grid

jguess north-south dimension of first guess grid

wndwgt adjustment factor for first guess winds.

minwnd minimum allowable wind speed on the agrmet grid

fg_hgt array of first guess isobaric heights

fg_rh array of first guess isobaric relative humidity

fg_tmp array of first guess isobaric temperatures

fg_wspd array of first guess surface wind speeds

kprs number of isobaric levels where fldblk needs first guess data

prslvls the isobaric levels where fldblk needs first guess data

alert_number counts number of alert messages sent

cstat I/O status, character

message Error message

count_dpd counts number of isobaric dewpoint depression levels read in from first guess file

count_hgt counts number of isobaric height levels read in from first guess file

count_rh counts number of isobaric relative humidity levels read in from first guess file

count_tmp counts number of isobaric temperature levels read in from first guess file

count_uwnd counts number of isobaric u wind levels read in from first guess file

count_vwnd counts number of isobaric v wind levels read in from first guess file

file_age stores forecast hour (0 for analysis) of first guess data

i,j,k looping and indexing variables
ierr,istat1 error status
ksec1 array of grib section 1 information
nunit unix file descriptor
calcrh function to calculate relative humidity
dum2d dummy array
fg_dpd array of first guess dewpoint depressions
fg_uwnd array of first guess surface u winds
fg_vwnd array of first guess surface v winds

The routines invoked are:

calcrh (29.1.37)
calculate relative humidities

29.1.37 calcrh (Source File: calcrh.F90)

REVISION HISTORY:

```
10 apr 89 initial version.....msgt neill/sddc
11 feb 91 chgd the floor rh value to 0.1 (10%). also isolated
            the debug stmt and declared it, itf, and k
            .....mr moore/sddc
04 mar 99 chgd the floor rh value back to 0.0 and increased the
            scope of the 2nd element of the array 'v' to -100:100
            to match the rhbds block data stmt.....mr moore/dnxm
14 jul 99 ported to IBM SP-2. pass in vapor pressure values
            thru the rhbds include file instead of thru a
            common block.....mr gayno/dnxm
20 aug 05 Sujay Kumar, Initial Specification in LIS
```

INTERFACE:

```
function calcrh ( temp, dpd )
implicit none
```

ARGUMENTS:

```
real, intent(in)      :: temp
real, intent(in)      :: dpd
```

DESCRIPTION:

to calculate relative humidities given temperatures and dew point depressions.

29.1.38 Method

- convert temps to kelvin and calc a dew pt temp - if temps are within reasonable range - derive look-up table indices for getting ambient and saturation vapor pressures, then get them. - calc rh by dividing ambient vapor pressure by saturation vapor pressure. keep value btwn 0 and 1 - if temps are not in reasonable range, set rh to -1

The arguments are:

temp temperature

dpd dew point temperature depression

calcrh relative humidity

29.1.39 fg_to_agr (Source File: fg_to_agr.F90)

REVISION HISTORY:

```
03 aug 99 initial version.....mr gayno/dnxm
12 aug 02 changed to four point interpolation from nearest
           neighbor approach.....mr gayno/dnxm
20 aug 05 Sujay Kumar, Initial Specification in LIS
```

INTERFACE:

```
subroutine fg_to_agr( fg_hgt, fg_rh, fg_tmp, fg_wspd,&
    ilon, jlat, hemi, imax, jmax, kprs, &
    ifguess, jfguess,&
    agr_tmp, agr_hgt, agr_rh, agr_wspd )
```

```
implicit none
```

ARGUMENTS:

```
integer,      intent(in)    :: hemi
integer,      intent(in)    :: ifguess
integer,      intent(in)    :: imax
integer,      intent(in)    :: jfguess
integer,      intent(in)    :: jmax
integer,      intent(in)    :: kprs

real,         intent(out)   :: agr_hgt  ( imax, jmax, kprs )
real,         intent(out)   :: agr_rh    ( imax, jmax, kprs )
real,         intent(out)   :: agr_tmp   ( imax, jmax, kprs )
real,         intent(out)   :: agr_wspd  ( imax, jmax )
real,         intent(in)    :: fg_hgt   ( ifguess, jfguess, kprs )
real,         intent(in)    :: fg_rh    ( ifguess, jfguess, kprs )
real,         intent(in)    :: fg_tmp   ( ifguess, jfguess, kprs )
real,         intent(in)    :: fg_wspd  ( ifguess, jfguess )
real,         intent(in)    :: ilon     ( imax, jmax, 2 )
real,         intent(in)    :: jlat     ( imax, jmax, 2 )
```

DESCRIPTION:

interpolate first guess data to the agrmet grid.

Method

interpolate geopotential heights, temperature, relative humidity, and wind speed to the agrmet grid using a four point interpolation.

The arguments are:

fg_hgt array of first guess isobaric heights

fg_rh array of first guess isobaric relative humidity

fg_tmp array of first guess isobaric temperatures

fg_wspd array of first guess surface wind speeds

ilon j coordinate of the agrmet grid points with respect to the first guess grid

jlat i coordinate of the agrmet grid points with respect to the first guess grid

hemi hemisphere (1=nh, 2=sh)

imax east-west dimension of agrmet grid

jmax north-south dimension of agrmet grid

kprs number of isobaric levels read in from the first guess data

ifguess i-dimension of the first guess grid

jfguess j-dimension of the first guess grid

agr_hgt geopotential heights on the agrmet grid

agr_rh relative humidity on the agrmet grid

agr_tmp temperature on the agrmet grid

agr_wpsd wind speed on the agrmet grid

i,j,k loop indices

xll x-coordinates of lower left corner of interpolation box

xlr x-coordinates of lower right corner of interpolation box

xul x-coordinates of upper left corner of interpolation box

xur x-coordinates of upper right corner of interpolation box

yll y-coordinates of lower left corner of interpolation box

ylr y-coordinates of lower right corner of interpolation box

yul y-coordinates of upper left corner of interpolation box

yur y-coordinates of upper right corner of interpolation box

lower,upper intermediate values in four point interpolation

29.1.40 ffill (Source File: ffill.F90)

REVISION HISTORY:

```
7 oct 97 initial version.....ssgt mccormick, mr moore/sysm
28 feb 99 changed array structure to supergrid. corrected
          temporal interpolation. removed equivalencing and
          simplified array structures. eliminated z6 and z6m
          .....mr moore/dnxm
22 Jul 99 ported to IBM SP-2. added intent attributes to all
          arguments. changed to grid specific arrays to
          be dynamically allocatable.....mr gayno/dnxm
01 aug 02 added read of data_levels namelist.....mr gayno/dnxm
```

INTERFACE:

```
subroutine ffill( alt, sfctmp, sfcrlh, sfcspd, sfcpes, land, &
                 jul, lastmp, lasrlh, lasspd, lasprs, &
                 step, order, hemi, minwnd, imax, jmax, &
                 kprs1, agr_tmp_c, agr_hgt_c, agr_rh_c, agr_wspd_c,&
                 agr_tmp_p, agr_hgt_p, agr_rh_p, agr_wspd_p, pathfb)
```

USES:

```
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

integer, intent(in)	:: imax
integer, intent(in)	:: jmax
integer, intent(in)	:: order
real, intent(in)	:: alt (imax,jmax)
real, intent(inout)	:: sfcpes (6,imax,jmax)
real, intent(inout)	:: sfcrlh (6,imax,jmax)
real, intent(inout)	:: sfcspd (6,imax,jmax)
real, intent(inout)	:: sfctmp (6,imax,jmax)
integer, intent(in)	:: land (imax,jmax)
integer, intent(in)	:: jul
real, intent(inout)	:: lasprs (imax,jmax)
real, intent(inout)	:: lasrlh (imax,jmax)
real, intent(inout)	:: lasspd (imax,jmax)
real, intent(inout)	:: lastmp (imax,jmax)
integer, intent(inout)	:: step
integer, intent(in)	:: hemi
real, intent(in)	:: minwnd
integer, intent(in)	:: kprs1
real, intent(in)	:: agr_hgt_c (imax,jmax, kprs1)
real, intent(in)	:: agr_rh_c (imax,jmax, kprs1)
real, intent(in)	:: agr_tmp_c (imax,jmax, kprs1)
real, intent(in)	:: agr_wspd_c (imax,jmax)
real, intent(in)	:: agr_hgt_p (imax,jmax, kprs1)
real, intent(in)	:: agr_rh_p (imax,jmax, kprs1)

```

real,      intent(in)      :: agr_tmp_p  ( imax,jmax, kprs1 )
real,      intent(in)      :: agr_wspd_p ( imax,jmax )
character(len=*), intent(in) :: pathfb

```

DESCRIPTION:

to determine a first guess field for the current time which will later be blended with observations to make a final analysis.

Method

- the first time through the 6-hour cycle (t+1)
- read data_levels namelist to get the isobaric levels for the first guess data.
- retrieve the first guess data for the end of the 6 hour cycle (t+6). (e.g. 12z data for a 12z run.)
- retrieve the first guess data for the start of the 6 hour cycle (t+0). (e.g. 06z data for a 12z run.) - time interpolate to the current julian hour using the retrieved first guess data from t+0 and t+6.
- for all other hours in the 6-hour cycle (t+2 to t+6):
- time interpolate to the current hour using the barnes analysis from the previous hour and first guess data from the end of the 6 hour cycle.

The arguments and variables are:

alt terrain elevation for the agrmet grid

land array of land mass points for agrmet grid

jul current julian hour

sfcprs first guess pressure fields on the AGRMET grid interpolated to the current time

sfcrlh first guess rh fields on the AGRMET grid interpolated to the current time

sfcspd first guess wind speed fields on the AGRMET grid interpolated to the current time

sfctmp first guess temperature fields on the AGRMET grid interpolated to the current time

lasprs first guess pressure fields for ending 6 hour time

lasrlh first guess rh fields for ending 6 hour time

lasspd first guess wind speed fields for ending 6 hour time

lastmp first guess temperature fields for ending 6 hour time

step time loop counter used to calculate time weights

order flag to indicate which data is to be read

hemi hemisphere (1=nh, 2=sh)

minwnd minimum allowable windspeed on the agrmet grid

imax east/west dimension of agrmet grid

jmax north/south dimension of agrmet grid

pathfb directory path of the surface calculations

kprs1 number of isobaric levels

agr_tmp_c temperature on the AGRMET grid for the current time

agr_hgt_c geopotential heights on the AGRMET grid for the current time
agr_rh_c relative humidity on the AGRMET grid for the current time
agr_wspd_c wind speeds on the AGRMET grid for the current time
agr_tmp_p temperature on the AGRMET grid from 6 hours ago
agr_hgt_p geopotential heights on the AGRMET grid from 6 hours ago
agr_rh_p relative humidity on the AGRMET grid from 6 hours ago
agr_wspd_p wind speeds on the AGRMET grid from 6 hours ago
action file i/o action - reading or opening
ctime character equivalent of variable time
istat file I/O status
i loop counter
julm6 current julian time of run minus 6 hours
nmlfil name, including path, of data_levels namelist
prslvls isobaric levels in fldbld data
time zulu hour of data_levels file
wt1,wt2 weight factors used for time smoothing prior hour's first guess data

The routines invoked are:

sfcval (29.1.41)
 vertically interpolate the first guess fields

29.1.41 sfcval (Source File: sfcval.F90)

REVISION HISTORY:

10 Apr 1989	Initial version.....MSgt Neill/SDDC
11 Feb 1991	added relative humidity value ceiling (100%) and floor (10%). added declaration for i and jMr Moore/SDDC
08 Oct 1997	variables jul, adr, data, fldbld, z6, and chour6 added to argument list so retrieval of 9 parameters could be moved here from driver. variables tmp10, tmp85, tmp70, dval10, dval85, dval70, relh10, and relh85 deleted from argument list since equivalencing of data array to these variables was moved here from driver. These variables also became local variables. Brought code up to current afgwc and sysm standards..Mr Moore(AGROMET), SSgt McCormick/SYSM
04 Mar 1999	changed to supergrid structure. corrected linear interpolation error. removed julian hour check for each box. Added vertical interpolation of relative humidity. Set consistant flagging of water values for

proper use by barnes. modified code for opening and
 reading fldblk files.....capt andrus, mr moore/dnxm
 11 Aug 1999 minor bug fixes during port to IBM SP-2. added
 intent attributes to arguments. made grid specific
 variables dynamically allocatable. removed
 nogaps conversion of dewpoint depression to
 relative humidity as this is now done in module
 fldblk. pass in field build directory path
 instead of hardwiring it here.....mr gayno/dnxm
 01 Aug 2002 modified routine to be more generic so it could
 handle any number of isobaric levels....mr gayno/dnxm

INTERFACE:

```

subroutine sfcval( alt, hemi, jul, land, minwnd, sfcprs, sfcrlh, &
  sfcspd, sfctmp, imax, jmax, &
  kprs, prslvls, kprs1,tmp, hgt, rlh, wndspd)
implicit none
  
```

ARGUMENTS:

integer, intent(in)	:: kprs
integer, intent(in)	:: kprs1
integer, intent(in)	:: imax
integer, intent(in)	:: jmax
real, intent(in)	:: alt (imax, jmax)
real, intent(in)	:: hgt (imax, jmax, kprs1)
real, intent(in)	:: rlh (imax, jmax, kprs1)
real, intent(in)	:: tmp (imax, jmax, kprs1)
real, intent(in)	:: wndspd (imax,jmax)
integer, intent(in)	:: hemi
integer, intent(in)	:: jul
integer, intent(in)	:: land (imax, jmax)
integer, intent(in)	:: prslvls (30)
real, intent(in)	:: minwnd
real, intent(out)	:: sfcprs (imax,jmax)
real, intent(out)	:: sfcrlh (imax,jmax)
real, intent(out)	:: sfcspd (imax,jmax)
real, intent(out)	:: sfctmp (imax,jmax)

DESCRIPTION:

vertically interpolate input fldblk fields to the model surface

Method

- allocate grid specific arrays.
- set file names and retrieve all input fldblk parameters
- over land, vertically interpolate isobaric fldblk data to the model surface. nogaps dewpoint depression was converted to relative humidity in module fldblk.
- if model terrain height is below the highest pressure level, set surface temperature, pressure and relative humidity to the values at that highest level.
- if model terrain height is above the lowest pressure level, set surface temperature, pressure and relative humidity to the values at that lowest pressure
- if model terrain height is between the pressure levels, vertically interpolate temperature and relative humidity. surface pressure is estimated using the hypsometric equation.

- surface winds are not vertically interpolated, they are simply set to the values calculated in module fidbld.
- range check all data.
- deallocate grid specific arrays.

The arguments and variables are:

alt terrain elevation for the agrmet grid
hemi hemisphere (1=nh,2=sh)
jul current julian hour
land array of land mass points for agrmet grid
minwnd minimum allowable windspeed on the agrmet grid
sfcprs first guess pressure fields on the AGRMET grid interpolated to the current time
sfcrlh first guess rh fields on the AGRMET grid interpolated to the current time
sfcspd first guess wind speed fields on the AGRMET grid interpolated to the current time
sfctmp first guess temperature fields on the AGRMET grid interpolated to the current time
imax east/west dimension of agrmet grid
jmax north/south dimension of agrmet grid
kprs1 number of isobaric levels
prslvls isobaric levels in first guess data
tmp temperature on the agrmet grid
hgt geopotential heights on the agrmet grid
rlh relative humidity on the agrmet grid
wndspd wind speeds on the agrmet grid
bashgt height of bottom of interpolation layer
basp pressure of bottom of interpolation layer
basrh relative humidity of bottom of interpolation layer
bastmp temperature at bottom of interpolation layer
calprs function for calculating surface pressure on the agrmet grid
h1 function that calculates $(R * T) / g$ which is used to hypsometrically calculate surface pressure on the agrmet grid
linint function for vertically interpolating isobaric data to the agrmet grid
thick thickness of interpolation layer
toprh relative humidity of top of interpolation layer
toptmp temperature of top of interpolation layer
ttop temperature of top of interpolation layer

29.1.42 getsfc (Source File: getsfc.F90)

REVISION HISTORY:

09 Jul 94	Initial version.....Capt Bertone, SYSM/AGROMET
06 Dec 95	Changed the second call of offhr2 to call original routine offhr1.....SSgt Erkkila/SYSM
12 Sep 97	Eliminated calls to onhour, offhr1 and offhr2. Converted from 6-hour to 1-hour retrieval of all observations. Brought code up to AFGWC & SYSM software standards.....Mr Moore(AGROMET), SSgt McCormick/SYSM
28 Feb 99	Modified to process data by hemisphere instead of by box, and to pull data from CDMS, to port the program from System 5 to GTWAPS (workstation).....Mr Moore/DNXM
21 Ful 99	Ported to IBM SP-2. Fixed some bugs. Revamped observation quality control logic. Removed hardwired AGRMET grid dimensions.....Mr Gayno/DNXM
30 Dec 99	Fixed error in call to agr_alert - the first argument was specified incorrectly.....Mr Gayno/DNXM
19 Jul 04	Added tracking of CDMS access time.....Mr Lewiston/DNXM

INTERFACE:

```
subroutine getsfc( julhr, ri, rj, obstmp, obsrlh, obsspd, obscnt, &  
    isize, hemi, minwnd, alert_number, imax, jmax, cdmsdir)
```

USES:

```
use listime_mgr, only : julhr_date  
use lis_logmod, only  : logunit, lis_alert  
  
implicit none
```

ARGUMENTS:

integer, intent(in)	:: julhr
integer, intent(in)	:: imax
integer, intent(in)	:: jmax
integer, intent(in)	:: isize
real, intent(out)	:: ri (isize)
real, intent(out)	:: rj (isize)
real, intent(out)	:: obsrlh (isize)
real, intent(out)	:: obsspd (isize)
real, intent(out)	:: obstmp (isize)
integer, intent(out)	:: obscnt
integer, intent(in)	:: hemi
integer, intent(inout)	:: alert_number
real, intent(in)	:: minwnd
character(len=*)	:: cdmsdir

DESCRIPTION:

Retrieve observed temperature, relative humidity, and wind speed values from the CDMS database.

Method

- Set observation counter to zero.

- Retrieve all observations for this hemisphere and time from database.
- If there is a read error, send an alert message. The observation count will remain at zero and no Barnes Analysis will be performed.
- Check lat/lon of each observation for missing data and bad values. Observations that fail check are not stored in the ob arrays passed to routine barnes.
- Convert from lat/lon to i and j and hemisphere on the AGRMET model grid.
- Check for observations located outside of grid.
- Check for observations converted to other hemisphere.
- Observations that fail either check are not stored to the ob arrays.
- Check for observations with missing/unreported temperatures, wind speeds and relative humidities. Don't store to ob arrays if it fails check.
- Descale temperature, wind speed and relative humidity for each observation. Range check data. If temperature or relative humidity is outside the range, store to ob arrays arrays as minus one so the Barnes Analysis will ignore it. If wind speed is out of range, constrain it to between 75 ms-1 and the value of minwnd. If any of the fields are missing/unreported, store as minus 1.
- If all three fields are missing, unreported and/or out of range, don't store this observation to the ob array.
- If the observation passes all the above checks, store the information to the final ob arrays for use in routine barnes and increment the observation counter.

The arguments and variables are:

julhr current julian hour

ri Array of observation locations on the AGRMET grid (i dimension)

rj Array of observation locations on the AGRMET grid (j dimension)

obstmp Array of temperature observations that have passed all quality control checks

obsrlh Array of relative humidity observations that have passed all quality control checks

obsspd Array of wind speed observations that have passed all quality control checks

obscnt Number of observations that have passed all quality control checks

isize Max number of observations allowed for a hemisphere

minwnd minimum allowable windspeed on the agrmet grid

alert_number incremental number of alert message

imax east/west dimension of agrmet grid

jmax north/south dimension of agrmet grid

cdmsdir full path of the CDMS directory

cjulhr character equivalent of variable julhr

hem Hemisphere returned from lltops routine

hemi Hemisphere (1 = north, 2 = south)

ierr1,ierr2,ierr3,istat1 I/O error status

ilat scaled observation latitude (integer)

ilon scaled observation longitude (integer)

irecord loop index for raw observation arrays

irelh scaled relative humidity observations (integer)

ispd Scaled wind speed observations (integer)
itmp Scaled temperature observations (integer)
norsou Character string for northern/southern hemisphere
nsize Number of observations returned from database
rlat descaled observation latitude
rlon descaled observation longitude
rrelh descaled observation relative humidity
rrspd descaled observation wind speed
rtmp descaled observation temperature
The routines invoked are:
julhr_date (5.4.23)
 convert julian hr to a date format
getsfcobsfilename (29.1.43)
 generates the surface OBS filename
lltops (7.0.24)
 convert latlon to points on the AGRMET grid
lis_alert (5.24.3)
 send an alert message with problems in OBS processing

29.1.43 getsfcobsfilename (Source File: getsfc.F90)

INTERFACE:

```
subroutine getsfcobsfilename(filename, dir, hemi, yr,mo,da,hr)
```

```
    implicit none
```

ARGUMENTS:

```
    character(len=*)      :: filename
    character(len=*)      :: dir
    integer, intent(in)   :: yr,mo,da,hr, hemi
```

DESCRIPTION:

This routines generates the name of the surface obs file
The arguments are:

hemi index of the hemisphere (1-NH, 2-SH)

dir full path to the directory containing the data

filename created filename

yr 4 digit year

mo integer value of month (1-12)

da day of the month

hr hour of the day

29.1.44 sfcalc_barnes (Source File: sfcalc_barnes.F90)

REVISION HISTORY:

```
28 feb 99 initial version.....capt andrus, mr moore/dnxm
22 jul 99 ported to IBM SP-2. added intent attributes to
           arguments. made all grid specific variables
           dynamically allocatable.....mr gayno/dnxm
```

INTERFACE:

```
subroutine sfcalc_barnes( obscnt, obs, ri, rj, fguess, radius, cparam, &
                        land, minwnd, isize, imax, jmax )

implicit none
```

ARGUMENTS:

integer,	intent(in)	:: isize
integer,	intent(in)	:: imax
integer,	intent(in)	:: jmax
integer,	intent(in)	:: obscnt
real,	intent(in)	:: obs (isize)
real,	intent(in)	:: ri (isize)
real,	intent(in)	:: rj (isize)
real,	intent(inout)	:: fguess (imax, jmax)
integer,	intent(in)	:: radius (imax, jmax)
character*2,	intent(in)	:: cparam
integer,	intent(in)	:: land (imax, jmax)
real,	intent(in)	:: minwnd

DESCRIPTION:

barnes optimal interpolation technique subroutine
adjust the first guess with observations using the barnes optimal analysis technique.

Method

- set limits for tossing bad observations (there are different limits depending upon the type of data.)
- allocate grid specific arrays.
- initialize weight and adjustment arrays to zero.
- loop through the observations.
 - for good observations (not previously flagged as bad in routine getsfc).
 - bilinearly interpolate the first guess value to the observation location and determine the difference between the observation and the interpolated first guess value (the residual).
 - for residuals within the acceptable limit, set the scan radius around the current observation.
 - loop through all grid points within the scan radius around the observation.
 - for land points, calculate the observation weight. calculate the observation's adjustment of the first guess and add it the total adjustment from all observations.
 - loop through all grid points
 - if observations have adjusted the first guess, calculate the final blended analysis.
 - range check the final analysis.
 - deallocate grid specific arrays.

The arguments and variables are:

obscnt Number of observations that have passed all quality control checks

obs array of observations

ri Array of observation locations on the AGRMET grid (i dimension)

rj Array of observation locations on the AGRMET grid (j dimension)

fguess on input - the first guess fields on output - the final barnes analysis

radius radius of influence at each agrmet grid point

cparam parameter to be processed (2 character code) ws - wind speed tp - temperature rh - relative humidity

land land/sea mask of agrmet grid

minwnd minimum allowable windspeed on the agrmet grid

isize Max number of observations allowed for

imax east/west dimension of agrmet grid

jmax north/south dimension of agrmet grid

a,b,c,d variables that hold the values of the first guess at the 4 agrmet grid points which surround the observation

del sum of the product of the observation weights and the residuals at an agrmet grid point

diff array of residuals, or the difference between the observation and the first guess at the observation point

diflim maximum allowable residual that is included in the barnes analysis

e,f intermediate values in the bilinear interpolation of the first guess to the observation point

maxi upper bound (i coordinate) of box in which barnes analysis is performed for one observation

maxj upper bound (j coordinate) of box in which barnes analysis is performed for one observation

mini lower bound (i coordinate) of box in which barnes analysis is performed for one observation

minj lower bound (j coordinate) of box in which barnes analysis is performed for one observation

ptrad distance between observations and agrmet grid point in grid lengths

sr radius of influence at an agrmet grid point

sumsqr function which calculates the sum of the square of the distance between an observation and the agrmet grid point in grid lengths

wt array of observation weights

wgt sum of the observation weights at an agrmet grid point

x the fractional portion of variable ri

y the fractional portion of variable rj

29.1.45 interp_agrmetvar (Source File: interp_agrmetvar.F90)

INTERFACE:

```
subroutine interp_agrmetvar(n,ip,gi,udef,varfield)
```

USES:

```
use lisdrv_module, only : lis
use agrmetdomain_module, only : agrmet_struc
implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: n
integer, intent(in)      :: ip
real, intent(in)         :: gi(2,agrmet_struc(n)%imax,agrmet_struc(n)%jmax)
real, intent(in)         :: udef
real, intent(inout)      :: varfield(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine interpolates a given AFWA field to the LIS domain (from polar stereographic grid to the LIS grid)

The arguments are:

n index of the nest

ip interpolation option

gi input AGRMET field (both hemispheres)

udef undefined value in the input field

varfield interpolated field in the LIS grid

The routines invoked are:

bilinear_interp (7.0.3)

spatially interpolate the forcing data using bilinear interpolation

neighbor_interp (7.0.7)

spatially interpolate the forcing data using neighbor interpolation

29.1.46 agrmet_fillgaps (Source File: agrmet_fillgaps.F90)

INTERFACE:

```
subroutine agrmet_fillgaps(n,ip,varfield)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use agrmetdomain_module, only : agrmet_struc
use lis_logmod, only    : logunit
implicit none
```

USES:

```
integer, intent(in)    :: n
integer, intent(in)    :: ip
real, intent(inout)   :: varfield(lis%lnc(n),lis%lnr(n))
```

DESCRIPTION:

This subroutine fills in invalid grid points introduced due to reprojection from PS to lat/lon. This routine assumes that the undef or invalid value is the LIS undefined value.

The arguments are:

n index of the nest

ip interpolation option

varfield updated output field

29.1.47 agrmet_svp (Source File: agrmet_svp.F90)

REVISION HISTORY:

```
30 jul 99 initial version.....lt col (ima) ken mitchell/dnxm
07 sep 99 ported to ibm sp-2. updated prolog. added intent
                  attributes to arguments.....mr gayno/dnxm
01 aug 05 Sujay Kumar, Adopted in LIS
```

INTERFACE:

```
subroutine agrmet_svp( qs, es, p, t )
```

```
implicit none
```

ARGUMENTS:

```
real, parameter          :: cpv = 1870.0
real, parameter          :: cw = 4187.0
real, parameter          :: eso = 611.2
real, parameter          :: rv = 461.5
real, parameter          :: to = 273.15

real, intent(out)        :: es
real, intent(in)         :: p
real, intent(out)        :: qs
real, intent(in)         :: t
```

DESCRIPTION:

calculates saturation vapor pressure and saturation specific humidity

Method

- call function e to calculate saturation vapor pressure
- calculate saturation specific humidity based on saturation vapor pressure

The arguments are:

e saturation vapor pressure function

es output saturation vapor pressure (pascals)

p input air pressure (pascals)

qs output saturation mixing ratio (kg kg⁻¹)

t input air temperature (k)

29.1.48 agrmet_longwv (Source File: agrmet_longwv.F90)

REVISION HISTORY:

```
15 may 88 initial version.....capt rice/sddc
07 sep 99 ported to ibm sp-2. added intent attributes to
                  arguments. updated prolog.....mr gayno/dnxm
29 jul 05 Sujay Kumar, Adopted in LIS
```

INTERFACE:

```
subroutine agrmet_longwv( sfctmp, e,iclamt, rldown )
implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: iclamt  ( 3 )
real,   intent(in)       :: e
real,   intent(out)      :: rldown
real,   intent(in)       :: sfctmp
```

DESCRIPTION:

to compute the net downward longwave radiation at the earth's surface.

Method

- calculate the emissivity of the clear sky.
- calculate downwelling longwave radiation of the clear sky.
- add contribution of low, middle and high clouds to the clear sky portion.

References:

dr idso's paper in the j. of geophys. research, no 74, pp 5397-5403.

dr r.f.wachtmann's paper in the digest of preprints, topical meeting on remote sensing of the atmosphere, anaheim,ca, optical society of america, entitled, "expansion of atmospheric temperature-moisture profiles in empirical orthogonal functions for remote sensing applications", 1975

The arguments and variables are:

e sfc vapor pressure (pascals)

iclamt low, mid, and high cloud amounts in percent

rldown net downward longwave irradiance.

sfctmp sfc temperature (deg k)
cldfrt fraction of low, mid, and high cloud amounts
clrsky irradiance contribution from the clear sky
emb sfc vapor pressure (millibars)
emissa idso clr sky emissivity (all wavelengths)
emissb sasc clr sky emissivity (adjusted emissa value)
hcterm high cloud emissivity term
lcterm low cloud emissivity term
mcterm mid cloud emissivity term
sigma stefan boltzman constant
zh high cloud height coefficient
zl low cloud height coefficient
zm mid cloud height coefficient

29.1.49 agrmet_calc_albedo (Source File: agrmet_calc_albedo.F90)

REVISION HISTORY:

```

20 oct 99 initial version based on ncep routines.....  

.....lt col mitchell/mr gayno/dnxm  

12 apr 02 pass in variables salp and snup instead of hard-wiring  

them. modified for new vegetation type database (USGS)  

.....mr gayno/dnxm  

14 may 02 initialized albedo array to zero.....mr gayno/dnxm  

8 aug 2005: Sujay Kumar; Initial Specification

```

INTERFACE:

```

subroutine agrmet_calc_albedo ( alb, albedo, shdfac, snoalb, &  

sneqv, snup, salp, bare)  
  

implicit none

```

ARGUMENTS:

real,	intent(in)	:: alb
real,	intent(out)	:: albedo
real,	intent(in)	:: salp
real,	intent(in)	:: shdfac
real,	intent(in)	:: sneqv
real,	intent(in)	:: snoalb
real,	intent(in)	:: snup
logical,	intent(in)	:: bare

DESCRIPTION:

calculate the snow-free albedo for the current day and then modify for current snow cover.

Method

- determine the current julian day.
- determine the seasonal snow-free albedo files that bound the current day.
- time interpolate snow-free albedo to the current day.
- calculate albedo based on snow-free value and the current snow cover.
- snow-free and snow-adjusted albedo are passed back to driver for use in ncep land-sfc model.

The arguments and variables are:

alb snow-free albedo on the agrmet grid for the current day

albedo albedo modified for snow cover (if any) for the agrmet grid for the current day

rsnow ratio of sneqv to snup

shdfac greenness fraction

sneqv snow liquid equivalent

snoalb maximum snow albedo

snofac snow-depth factor

snup threshold snow depth (in water equivalent m) that implies 100

29.1.50 agrmet_tr_coeffs (Source File: agrmet_tr_coeffs.F90)

REVISION HISTORY:

```
10 feb 88 initial version.....capt rice/sddc
21 apr 89 testing, updating, error corrections....rice&moore/sddc
07 sep 99 ported to ibm-sp2. updated prolog. removed all
          8th mesh "box" logic. replaced call to grd2ll with
          new utility pstoll.....mr gayno/dnxm
10 jun 02 removed all references to rtneph.....mr gayno/dnxm
03 aug 05; Sujay Kumar, Adopted in LIS
```

INTERFACE:

```
subroutine agrmet_tr_coeffs( i,j,hemi, fog, icltyp, iclamt,    &
                           r1,r2,r3,t1,t2,t3,coszen,yr1,mo1,da1,hr1)
```

USES:

```
use constantsMod, only : CONST_PI, CONST_SOLAR
use listime_mgr, only : get_julhr
implicit none
```

ARGUMENTS:

```
integer,      intent(in)      :: iclamt ( 3 )
integer,      intent(in)      :: icltyp ( 3 )
logical,      intent(in)      :: fog
integer,      intent(in)      :: hemi
integer,      intent(in)      :: i
```

```

integer,      intent(in)    :: j
real          :: r1
real          :: r2
real          :: r3
real          :: t1
real          :: t2
real          :: t3
real          :: coszen
integer        :: yr1,mo1,da1,hr1

```

DESCRIPTION:

to compute the transmissivity and reflectivity coefficients

Method

shapiro's method is based on a 3-layer plane-parallel atmosphere. each layer (high, middle, and low clouds) transmits and reflects some of the solar radiation incident on it from above and from below. each layer's trans- missivity and reflectivity value depends on the layer's cloud type and amount and on the solar zenith angle. empirically derived reflectivity and transmissivity values for each of the three layers and the fraction of the solar radiation entering the top of the troposphere (for the given location are used to calculate the surface insolation for the point.

The arguments and variables are:

alat latitude of point.

albedo surface albedo of the point

alon longitude of the point.

asolcn amount of solar radiation entering the top of the atmosphere for a particular time of the year.

cdel cosine of the sun's declination angle.

clat cosine of the sun's elevation angle.

cldamt fraction of low, middle, and high cloud amounts.

cldtyp low, middle, and high cloud types.

clrtyp shapiro low, middle, and high clear cloud types.

coszen solar zenith angle.

d2 double back scatter from air and clouds.

deltim longitudnal zulu time difference.

fog present weather that indicates fog is present

frac intermediate step to fracsq.

fracsq fraction of sunlight based on time of year.

hemi hemisphere (1=nh, 2=sh)

hrangl hour angle of the sun.

iclamt low, middle, and high cloud amounts.

ictyp low, middle, and high cloud types.

julday julian day of the calendar year.

pi geometric pi = 3.14.
pid12 coefficient for hour angle.
pid180 coefficient for sun' angle.
r1 reflectivity coefficient for high clouds.
r2 reflectivity coefficient for mid clouds.
r3 reflectivity coefficient for low clouds.
rtop shortwave radiation at the top of the atmosphere.
sdec solar declination angle
sdel solar zenith angle
slat sine of sun's elevation angle.
solcon solar constant (W m-2)
t1 transmissivity coefficient for high clouds.
t2 transmissivity coefficient for mid clouds.
t3 transmissivity coefficient for low clouds.
ztime current zulu time.
The routines invoked are:
agrmet_tpcnv (29.1.52)
 cloud type conversion routine
get_julhr (5.4.9)
 computes the current julian hour
agrmet_trnref (29.1.53)
 computes transmissivities
agrmet_bakfac (29.1.54)
 computes the backscatter factor

29.1.51 agrmet_solar (Source File: agrmet_solar.F90)

REVISION HISTORY:

```

10 feb 88 initial version.....capt rice/sddc
21 apr 89 testing, updating, error corrections....rice&moore/sddc
07 sep 99 ported to ibm-sp2. updated prolog. removed all
          8th mesh "box" logic. replaced call to grd2ll with
          new utility pstoll.....mr gayno/dnxm
10 jun 02 removed all references to rtnehp.....mr gayno/dnxm
03 aug 05; Sujay Kumar, Adopted in LIS

```

INTERFACE:

```

subroutine agrmet_solar ( coszen, albdo, &
                         r1,r2,r3,t1,t2,t3,rsolin,yr1,mo1,da1,hr1)

```

USES:

```
use constantsMod, only : CONST_PI, CONST_SOLAR
use listime_mgr, only : get_julhr

implicit none
```

ARGUMENTS:

real		:: coszen
real,	intent(in)	:: albdo
real,	intent(out)	:: rsolin
real		:: r1
real		:: r2
real		:: r3
real		:: t1
real		:: t2
real		:: t3
integer		:: yr1,mo1,da1,hr1

DESCRIPTION:

to compute the net downward shortwave (solar) radiation value at the earth's surface.

Method

shapiro's method is based on a 3-layer plane-parallel atmosphere. each layer (high, middle, and low clouds) transmits and reflects some of the solar radiation incident on it from above and from below. each layer's transmissivity and reflectivity value depends on the layer's cloud type and amount and on the solar zenith angle. empirically derived reflectivity and transmissivity values for each of the three layers and the fraction of the solar radiation entering the top of the troposphere (for the given location are used to calculate the surface insolation for the point.

The arguments and variables are:

alat latitude of point.

albedo surface albedo of the point

alon longitude of the point.

asolcn amount of solar radiation entering the top of the atmosphere for a particular time of the year.

cdel cosine of the sun's declination angle.

clat cosine of the sun's elevation angle.

coszen solar zenith angle.

d2 double back scatter from air and clouds.

deltim longitudnal zulu time difference.

fog present weather that indicates fog is present

frac intermediate step to fracsq.

fracsq fraction of sunlight based on time of year.

hemi hemisphere (1=nh, 2=sh)

hrangl hour angle of the sun.

iclamt low, middle, and high cloud amounts.
iclytyp low, middle, and high cloud types.
julday julian day of the calendar year.
pi geometric pi = 3.14.
pid12 coefficient for hour angle.
pid180 coefficient for sun' angle.
r1 reflectivity coefficient for high clouds.
r2 reflectivity coefficient for mid clouds.
r3 reflectivity coefficient for low clouds.
rsolin solar radiation (w m-2)
rtop shortwave radiation at the top of the atmosphere.
sdec solar declination angle
sdel solar zenith angle
slat sine of sun's elevation angle.
solcon solar constant (W m-2)
t1 transmissivity coefficient for high clouds.
t2 transmissivity coefficient for mid clouds.
t3 transmissivity coefficient for low clouds.
ztime current zulu time.

The routines invoked are:

agrmet_bakfac (29.1.54)
computes the backscatter factor

29.1.52 agrmet_typcnv (Source File: agrmet_typcnv.F90)

REVISION HISTORY:

10 feb 88	initial version.....	capt rice/sddc
21 apr 89	testing, logic improvements.....	rice&moore/sddc
07 sep 99	ported to ibm sp-2. added intent attributes to arguments. updated prolog.....	mr gayno/dnxm
10 jun 02	change all references from rtneph to cdfs2.....
03 aug 05	Sujay Kumar, Adopted in LIS	mr gayno/dnxm

INTERFACE:

```
subroutine agrmet_typcnv( icltyp, iclamt, fog, cldtyp, clrtyp, cldamt )
```

29.1.53 agrmet_trnref (Source File: agrmet_trnref.F90)

REVISION HISTORY:

```
15 feb 88 initial version.....rice,moore/sddc
21 apr 89 testing, logic improvements.....rice&moore/sddc
07 sep 99 ported to ibm sp-2. updated prolog. added intent
           attributes to arguments.....mr gayno/dnxm
03 aug 05; Sujay Kumar, Adopted in LIS
```

INTERFACE:

```
subroutine agrmet_trnref( cldtyp, cldamt, clrtype, coszen, &
                         t1, t2, t3, r1, r2, r3 )

implicit none
```

ARGUMENTS:

```
integer,      intent(in)    :: cldtyp  ( 3 )
integer,      intent(in)    :: clrtype ( 3 )
real,         intent(in)    :: cldamt   ( 3 )
real,         intent(in)    :: coszen
real,         intent(out)   :: r1
real,         intent(out)   :: r2
real,         intent(out)   :: r3
real,         intent(out)   :: t1
real,         intent(out)   :: t2
real,         intent(out)   :: t3
```

DESCRIPTION:

to compute all three (low, middle, and high cloud) transmissivities, and reflectivities.
Method

- decide if radiation incident on the layer in question is direct or diffuse (n/a for top layer).
- call trcalc routine to calculate the required values.

The arguments and variables are:

cldamt fraction of low, mid, and high cloud amounts.

cldtyp low, mid, and high cloud types.

clrtype shapiro low, mid, and high cloud types.

coszen solar zenith angle.

difuse logical operator for diffuse radiation.

level identifies cloud level (low, mid, or high).

r1 high level reflectivity coefficient.

r2 mid level reflectivity coefficient.

r3 low level reflectivity coefficient.

t1 high level transmissivity coefficient.

t2 mid level transmissivity coefficient.

t3 low level transmissivity coefficient.

The routines invoked are:

agrmet_trcalc (29.1.57)
computes tranmissivity and reflectivities

29.1.54 agrmet_bakfac (Source File: agrmet_bakfac.F90)

REVISION HISTORY:

```
15 feb 88 initial version.....rice/sddc
07 sep 99 ported to ibm sp-2. updated prolog. added intent
           attributes to arguments.....mr gayno/dnxm
03 aug 05; Sujay Kumar, Adopted in LIS
```

INTERFACE:

```
function agrmet_bakfac( t2, t3, r1, r2, r3, albdo )

implicit none
!ARGUMENTS
real,      intent(in)      :: albdo
real,      intent(in)      :: r1
real,      intent(in)      :: r2
real,      intent(in)      :: r3
real,      intent(in)      :: t2
real,      intent(in)      :: t3
real                  :: agrmet_bakfac
```

DESCRIPTION:

to compute the double backscattering effect that the air and clouds have on solar radiation.

Method

calculate the double backscatter effect (card2) based on the transmissivity and reflectivity coefficients of the sky and the surface albedo.

The arguments and variables are:

albedo surface albedo.

agrmet_bakfac overall backscattering effect.

capd1 intermediate step for backscatter equation.

capd2 the backscatter equation as described by shapiro.

capdo intermediate step for backscatter equation.

d1 intermediate step for backscatter equation as a function of reflectivity.

d2 intermediate step for backscatter equation as a function of reflectivity.

d3 intermediate step for backscatter equation as a function of reflectivity and albedo.

r1 reflectivity coefficient of low clouds.
r2 reflectivity coefficient of mid clouds.
r3 reflectivity coefficient of high clouds.
t2 transmissivity coefficient of mid clouds.
t3 transmissivity coefficient of high clouds.

29.1.55 agrmet_w (Source File: agrmet_w.F90)

REVISION HISTORY:

```
01 mar 88 initial version.....rice/sddc
25 apr 89 improve coeff and testing.....rice&moore/sddc
07 sep 99 ported to ibm sp-2. added intent attributes to
          arguments.....mr gayno/dnxm
03 aug 05; Sujay Kumar, Adopted in LIS
```

INTERFACE:

```
function agrmet_w( cldtyp, cldamt, level, coszen )

implicit none
```

ARGUMENTS:

```
integer, intent(inout)      :: cldtyp ( 3 )
integer, intent(in)         :: level
real,    intent(in)         :: cldamt     ( 3 )
real,    intent(in)         :: coszen
```

DESCRIPTION:

to calculate a weighting coefficient to be used in determining transmissivity and reflectivity values for partly cloudy conditions from clear and cloudy transmissivity and reflectivity values.

Method

- a weight is assigned based on the level of cloud types and amounts plus the effect of the solar zenith angle using shapiro's polynomials.
- this weight is then applied to the cloud amount at a particular level.

The arguments and variables are:

c0 pointers for weighting polynomial.
c1 pointers for weighting polynomial.
c2 pointers for weighting polynomial.
c3 pointers for weighting polynomial.
c4 pointers for weighting polynomial.
c5 pointers for weighting polynomial.
ca dummy cld amt variable

cldamt fraction of low, mid, and high cloud amounts.

cldtype low, mid, and high cloud types.

coef shapiro's polynomial coefficients.

coszen solar zenith angle.

level pointer for cloud level.

agrmet_w final weighting coefficient passed back to trcalc.

w1 intermediate weighting coefficient.

w2 intermediate weighting coefficient.

29.1.56 agrmet_trpoly (Source File: agrmet_trpoly.F90)

REVISION HISTORY:

```
01 mar 88    initial version.....capt rice/sddc
25 apr 89    testing and logic improvements.....rice & moore/sddc
07 sep 99    ported to ibm sp-2. updated prolog. added intent
              attributes to arguments.....mr gayno/dnxm
```

03 aug 05; Sujay Kumar, Adopted in LIS

INTERFACE:

```
subroutine agrmet_trpoly( cldtyp, clrtype, coszen, level, sky, trn, ref )
implicit none
```

ARGUMENTS:

```
integer,      intent(inout) :: cldtyp      ( 3 )
integer,      intent(in)   :: clrtype      ( 3 )
integer,      intent(in)   :: level
logical,      intent(in)   :: sky
real,         intent(in)   :: coszen
real,         intent(out)  :: trn
real,         intent(out)  :: ref
```

DESCRIPTION:

to compute the transmissivity and reflectivity coefficients using shapiro's method.

Method

- compute the cosine function for reflectivity and transmissivity, then the clear sky coefficient, first for transmissivity, then for reflectivity.
- repeat these steps for cloudy skies.

The arguments and variables are:

a0 reflectivity pointer for cloudy sky polynomial

a1 reflectivity pointer for cloudy sky polynomial

a2 reflectivity pointer for cloudy sky polynomial
a3 reflectivity pointer for cloudy sky polynomial
b0 transmissivity pointer for cloudy sky polynomial
b1 transmissivity pointer for cloudy sky polynomial
b2 transmissivity pointer for cloudy sky polynomial
b3 transmissivity pointer for cloudy sky polynomial
c0 trans/reflect argument for function fac
c1 trans/reflect argument for function fac
c2 trans/reflect argument for function fac
c3 trans/reflect argument for function fac
cldtype low,mid, and high cloud types
clrtyp low,mid, and high clear cloud types
cosz cosine of the solar zenith angle
coszen cosine of the solar zenith angle
level pointer for cloud level
rc clear sky reflectivity polynomial values
ref the reflectivity value sent back to trcalc
rhoc cloudy sky reflectivity polynomial values
sky flag for clear or cloudy sky
tauc cloudy sky transmissivity polynomial values
tc clear sky transmissivity polynomial values
trn final transmissivity value sent back to trcalc

29.1.57 agrmet_trcalc (Source File: agrmet_trcalc.F90)

REVISION HISTORY:

01 feb 88 initial version.....rice,moore/sddc
21 apr 89 testing and logic improvements.....rice&moore/sddc
07 sep 99 ported to ibm sp-2. updated prolog. added intent
attributes to arguments.....mr gayno/dnxm
03 aug 05; Sujay Kumar, Adopted in LIS

INTERFACE:

```

subroutine agrmet_trcalc( cldamt, cldtyp, clrtyp, coszen, level, difuse, &
    trans, reflec)

implicit none

integer, intent(in)      :: cldtyp  ( 3 )
integer, intent(in)      :: clrtyp  ( 3 )
integer, intent(in)      :: level
logical, intent(in)      :: difuse
real,   intent(in)       :: cldamt  ( 3 )
real,   intent(in)       :: coszen
real,   intent(out)      :: reflec
real,   intent(out)      :: trans

```

DESCRIPTION:

to calculate the transmissivity and reflectivity coefficients for low, mid, and high clouds at the level in question with a direct or diffuse insolation fm above.

Method

- compute clear sky diffuse transmissivity and reflectivity coefficients.
- if overcast, computes cloudy layer coefficients.
- if partly cloudy, compute the weighting factor for transmissivity and reflectivity. weight = 1.0 for diffuse sky. the non-diffuse sky coefficients are computed in trpoly.

The arguments and variables are:

cldamt fraction of low, mid, and high cloud amounts.

cldtyp low, mid, and high cloud types.

clrtyp shapiro low, mid, and high clear cloud types.

coszen cosize of solar zenith angle.

difuse logical identifier for diffuse sky.

level pointer for low, mid, and high cloud level.

phi equal to cloud amount times weighting coefficient.

rcldy reflectivity coefficients for cloudy sky.

rclr reflectivity coefficients for clear sky.

rdfs reflectivity coefficients for clear diffuse sky.

reflec final reflectivity coefficient to trnref.

rhodfs reflectivity coefficients for diffuse cloudy sky.

taudfs transmissivity coefficients for diffuse cloudy sky

tcldy transmissivity coefficients for cloudy sky.

tclr transmissivity coefficients for clear sky.

tdfs transmissivity coefficients for diffuse clear sky.

trans final transmissivity coefficient to trnref.

weight weight for non-diffuse sky.

The routines invoked are:

agrmet_trpoly (29.1.57)
transmissivity and reflectivity polynomial routine

agrmet_w (29.1.55)
computes weighting coefficient

29.1.58 loadcloud (Source File: loadcloud.F90)

REVISION HISTORY:

```
11 feb 91 initial version.....mr moore/sddc(agromet)
05 may 91 changed estwt arguments to thres, made thres an array
          deleted the assignment statements using estwt and the
          nint function, delted the estwt array, updated the
          prolog.....capt bertone/sddc(agromet)
01 jun 92 chgd range of scalar latbnd to match new size of array
          pcoef elsewhere in pgm. updtd prolog.....
.....mr moore/sysm(agromet)
02 sep 99 ported to ibm sp-2. removed "box" logic. updated
          prolog. added intent attributes to arguments.....
.....mr gayno/dnxm
10 jun 02 modified to use cdfs2 data.....mr gayno/dnxm
29Jul2005; Sujay Kumar, Adopted in LIS
```

INTERFACE:

```
subroutine loadcloud(hemi, land,thres, times, amounts, tops, types, &
cldtyp, cldamt, fog, julhr, imax,jmax)

implicit none
```

ARGUMENTS:

integer, intent(in)	:: imax
integer, intent(in)	:: jmax
byte, intent(in)	:: amounts (4, 1024, 1024)
byte, intent(in)	:: types (4, 1024, 1024)
integer, intent(in)	:: land (imax,jmax)
integer, intent(out)	:: cldamt (3, imax,jmax)
integer, intent(out)	:: cldtyp (3, imax,jmax)
logical, intent(out)	:: fog (imax,jmax)
integer, intent(in)	:: hemi
integer, intent(in)	:: julhr
integer, intent(in)	:: thres (5)
integer*4, intent(in)	:: times (1024, 1024)
integer*2, intent(in)	:: tops (4, 1024, 1024)

DESCRIPTION:

to convert cdfs2 data for use by the flux3 radiation routines.

Method

- check if the point is land.
- loop over all four cdfs2 layers. - check time of cdfs2 data, if within acceptable time range then
- convert from cdfs2 cloud type to flux3 cloud type.
- if cdfs2 says type is cb, see if cloud top is below user-specified threshold. if it is, set flux3 cloud type to cumulus instead.
- set flux3 cloud amount for its three layers to the maximum amount of low/mid/high clouds from cdfs2.
- if data is old, set flux3 cloud arrays to zero (clear skies).

The arguments and variables are:

amounts cdfs2 cloud amounts

cldamt output cloud amounts for use in radiation calcs

cldtim cdfs2 pixel times

cldtyp output cloud types for use in radiation calcs

fog fog present flag (logical)

hemi current hemisphere (n = 1, s = 2)

i loop counter

ii corresponding agrmet i-coordinate on the cdfs2 grid

ixx agrmet grid dimension, i-direction

j loop counter

jj corresponding agrmet j-coordinate on the cdfs2 grid

julhr current julian hour

jyy agrmet grid dimension, j-direction

land point processing switches (land-sea mask)

latbnd latitude band corresponding to cb thresholds

lonslc longitude slice

lvl loop counter (cdf2 cloud level)

nefamt cdfs2 cloud amount

neftop cdfs2 cloud top height

neftyp cdfs2 cloud type

thres cb cloud top thresholds from flux3 control file

times cdfs2 pixel times

tops cdfs2 cloud tops

types cdfs2 cloud types

The routines invoked are:

agrmet_bndslc (29.1.59)
determine latitude band and longitude slice

29.1.59 agrmet_bndslc (Source File: agrmet_bndslc.F90)

REVISION HISTORY:

```
09 jul 94 initial version.....capt bertone, sysm(agromet)
16 sep 99 ported to ibm sp2. removed box logic. added intent
           attributes to arguments. added call to utility
           routine pstoll, which replaces an obsolete utility
           routine grdtll.....mr gayno/dnxm
01 aug 05 Sujay Kumar, Adopted in LIS with modifications
```

INTERFACE:

```
subroutine agrmet_bndslc(i,j, hemi, latbnd, lonslc )

implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: hemi
integer, intent(out)     :: latbnd
integer, intent(out)     :: lonslc
integer, intent(in)      :: i
integer, intent(in)      :: j
```

DESCRIPTION:

to determine and return to the parent routine the latitude band and longitude slice into which this point falls.

Method

- find the band and slice into which this point's lat and lon fall.

The arguments and variables are:

hemi hemisphere (1 = north, 2 = south)

i i-coordinate of agrmet grid point

j j-coordinate of agrmet grid point

latbnd latitude band

lon longitude of agrmet point

lonslc longitude slice

ri real i-coordinate of agrmet grid point

rz real j-coordinate of agrmet grid point

The routines invoked are:

pstoll (7.0.25)
computes the lat lon values of a PS grid point

29.1.60 readagrmetpcpforcing (Source File: readagrmetpcpforcing.F90)

REVISION HISTORY:

29Jul2005; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine readagrmetpcpforcing(n,order)
```

USES:

```
use lisdrv_module, only      : lis,lisdom
use agrmetdomain_module, only : agrmet_struc
use baseforcing_module, only  : lisforc
use spmdMod
use listime_mgr, only        : julhr_date, get_julhr
use lis_logmod, only         : logunit
use lis_fileIOMod, only      : putget

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: order
```

DESCRIPTION:

This routine calls the AGRMET precipitation analysis for the current time. The analysis includes the use of observed precipitation (rain gauge reports) from AFWA's global surface observation database, estimates using methods based on remotely-sensed and climatological data. The analysis also blends real and estimated precip amounts using a barnes technique. A hierarchy of these estimates is established to ensure the most reliable data is used.

The arguments and variables are:

order flag indicating which data to be read (order=1, read the previous hourly instance, order=2, read the next hourly instance)

n index of the nest

hemi index of hemisphere loops

c,r,i,t looping and indexing variables

use_twelve flag to use the 12-hourly precip amounts or the 6 hourly amounts

julbeg starting julian hour

julend ending julian hour

j3hr 3 hourly julian time

p6 6 hourly rain-gauge precip amounts (mm) on the AGRMET grid

estpcp array of precip estimates (last two 3-hourly periods)

estp6 estimated 6hr precip array (mm/6hr)

source array that contains source-of-the-estimate information with specified source flags.

cdfs2est CDFS2 cloud based precip estimate (mm/3hr)

prcpwe present/past weather estimate on the grid

p12 12 hourly precip amounts (mm) on the AGRMET grid

mrgp6 6 hourly merged precip amounts (mm)

relp parsed 3 hour real precip array (mm/3hr)

relp6 parsed 6hr real precip array (mm/6hr)

cdfsii6 6hr CDFS2 based precip estimate (mm/6hr)

srcwts value weight placed on each source

addrad added radius value

maxrad maximum radius value

minrad minimum radius value

varrad radius change amount due to the real precip variance

tmp temporary array to hold weighted values

wgt weighted array point values

curr_time current time

pcp_process flag to determine if the precip analysis should be performed in the current hour

varfield interpolated variable

yr1,mo1,da1,hr1,mn1,ss1 time/date specific variables

alert_number alert message number

gi input merged precip field

quad9r undefined value

ip interpolation option

c,r,k looping and indexing variables

pathpcp Path to agrmet precip files

The routines invoked are:

find_agrppcp_starttime (29.1.61)
computes the start time for the precip processing.

julhr_date (5.4.23)
converts julian hour to a date format

getpcpobs (29.1.65)
read AFWA precip surface observations

makest (29.1.73)
create an consolidated precip amount based on several possible precipitation estimates

phsrel12 (29.1.83)
retrieve phase 12 hourly rain gauge amounts

phsrel6 (29.1.85)
retrieve phase 6 hourly raing gauge amounts

agrmet_valid (29.1.87)
validate precip amounts

pcp_barnes (29.1.88)
perform barnes analysis on the precip data

make03 (29.1.89)
make 3 hourly merged precipitation

interp_agrmetvar (29.1.45)
spatial interpolation of an AGRMET variable to LIS grid

agrmet_fillgaps (29.1.46)
fills the gaps in the interpolated field due to mismatches in LIS and AGRMET masks

29.1.61 find_agrpcp_starttime (Source File: `readagrmepcpforcing.F90`)

INTERFACE:

```
subroutine find_agrpcp_starttime(yr,mo,da,hr,julbeg)
```

USES:

```
use listime_mgr, only : tick, get_julhr  
  
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: yr  
integer, intent(in) :: mo  
integer, intent(in) :: da  
integer, intent(in) :: hr  
integer, intent(inout) :: julbeg
```

DESCRIPTION:

This routine finds the julian hour to start the AGRMET precip processing from, based on the current input time.

The arguments are:

yr the current year

mo the current month

da the current day

hr the current hour

julbeg output starting julian hour

The routines invoked are:

tick (5.4.22)
computes previous 6 hour time.

get_julhr (5.4.9)
converts the date to a julian hour

29.1.62 find_agrccp_readtime (Source File: readagrmetpcpforcing.F90)

INTERFACE:

```
subroutine find_agrccp_readtime(yr,mo,da,hr,julhr)
```

USES:

```
use listime_mgr, only : tick, get_julhr  
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: yr  
integer, intent(in) :: mo  
integer, intent(in) :: da  
integer, intent(in) :: hr  
integer, intent(inout) :: julhr
```

DESCRIPTION:

This routine finds the julian hour to read the AGRMET precip from, based on the current input time.
The arguments are:

yr the current year

mo the current month

da the current day

hr the current hour

julbeg output AGRMET precip reading time

The routines invoked are:

tick (5.4.22)
computes previous 6 hour time.

get.julhr (5.4.9)
converts the date to a julian hour

29.1.63 readagrmetpcpforcinganalysis (Source File: readagrmetpcpforcinganalysis.F90)

REVISION HISTORY:

29Jul2005; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine readagrmetpcpforcinganalysis(n,order)
```

USES:

```

use lisdrv_module, only      : lis,lisdom
use agrmetdomain_module, only : agrmet_struc
use baseforcing_module, only  : lisforc
use spmdMod
use listime_mgr, only        : julhr_date, get_julhr
use lis_logmod, only         : logunit
use lis_fileIOMod, only      : putget

implicit none

```

ARGUMENTS:

```

integer, intent(in) :: n
integer, intent(in) :: order

```

DESCRIPTION:

This routine calls the previously generated AGRMET precipitation analysis for the current time.
The arguments and variables are:

order flag indicating which data to be read (order=1, read the previous hourly instance, order=2, read the next hourly instance)

n index of the nest

hemi index of hemisphere loops

c,r,i,t looping and indexing variables

use_twelve flag to use the 12-hourly precip amounts or the 6 hourly amounts

julbeg starting julian hour

julend ending julian hour

varfield interpolated variable

yr1,mo1,da1,hr1,mn1,ss1 time/date specific variables

alert_number alert message number

gi input merged precip field

quad9r undefined value

ip interpolation option

c,r,k looping and indexing variables

The routines invoked are:

get_agrpccp_readtime (29.1.64)

computes the reading time for the precip processing.

julhr_date (5.4.23)

converts julian hour to a date format

interp_agrmetvar (29.1.45)

spatial interpolation of an AGRMET variable to LIS grid

agrmet_fillgaps (29.1.46)

fills the gaps in the interpolated field due to mismatches in LIS and AGRMET masks

putget (5.23.1)

reads the data from the previously generated analysis

29.1.64 get_agrppcp_readtime (Source File: readagrmepcpforcinganalysis.F90)

INTERFACE:

```
subroutine get_agrppcp_readtime(yr,mo,da,hr,mn,julhr)
```

USES:

```
use listime_mgr, only : tick, get_julhr  
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: yr  
integer, intent(in) :: mo  
integer, intent(in) :: da  
integer, intent(in) :: hr  
integer, intent(in) :: mn  
integer, intent(inout) :: julhr
```

DESCRIPTION:

This routine gets the julian hour to read the AGRMET precip from, based on the current input time.
The arguments are:

yr the current year

mo the current month

da the current day

hr the current hour

julbeg output AGRMET precip reading time

The routines invoked are:

tick (5.4.22)
computes previous 6 hour time.

get_julhr (5.4.9)
converts the date to a julian hour

29.1.65 getpcpobs (Source File: getpcpobs.F90)

REVISION HISTORY:

29Oct2005; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine getpcpobs(n, hemi, j6hr, month, prcpwe, &  
use_twelve, p6, p12, alert_number)
```

USES:

```

use lisdrv_module, only : lis
use agrmetdomain_module, only : agrmet_struc
use listime_mgr, only : tick, julhr_date
use lis_logmod, only : logunit

implicit none

```

ARGUMENTS:

```

integer, intent(in)    :: n
real,   intent(out)    :: prcpwe(agrmet_struc(n)%imax,agrmet_struc(n)%jmax,2)
real,   intent(inout)   :: p6(agrmet_struc(n)%imax,agrmet_struc(n)%jmax)
real,   intent(inout)   :: p12(agrmet_struc(n)%imax,agrmet_struc(n)%jmax)
logical, intent(in)    :: use_twelve
integer, intent(in)    :: hemi
integer, intent(in)    :: j6hr
integer, intent(in)    :: month
integer, intent(inout):: alert_number

```

DESCRIPTION:

This routine retrieves precip observations from the CDMS database.

Method

- Initialize necessary arrays.
- Call agrprc to get obs from CDMS database.
- Note: in the northern hemisphere, the database is called once for current Julian hour. In the southern hemisphere it is called multiple times - at the current Julian hour; one hour before and after the current Julian hour; and two hours before the current Julian hour. This is done because Australia does not report at the synoptic times but rather at local time (and the local reporting time changes with season. Confusing, huh?).
- Call getpwe to calculate rainfall based on current or past weather reports.
- At 00, 06, 12 and 18 utc, call storeobs to perform some preprocessing of the rain gauge data. then call processobs to put the rain gauge data on the grid.

The arguments and variables are:

alert_number number of alert messages
bsn WMO block station number array
duration Valid period of precipitation
endjul Ending Julian hour of the one-hour loop
hemi Hemisphere flag (1 = N, 2 = S)
ierr1,ierr2,ierr3 I/O error status
ilat Array of observation latitudes
ilon Array of observation longitudes
imax Grid dimension - east/west direction
isize Maximum array size of observation arrays
jmax Grid dimension - north/south direction
j1hr One-hourly loop index
j3hr Three-hourly loop index

j6hr Beginning of this 6 hourly processing period in Julian hours

k Loop counter

mscprc Miscellaneous precip amounts

month Current month

nsize Number of obs returned from database

obs Array of processed observations

amt6 Six hourly precip amount

amt12 Twelve hourly precip amount

amt24 Twenty-four hourly precip amount

lat Latitude

lon Longitude

wmonum WMO block station number

oldd_pwe Stores distances between a grid point and its nearest neighbor observation

pastwx Past weather at station

pathpcp Directory path of precip data directory

prcpwe Present/past weather estimate on the grid

preswx Present weather at station

p6 6-hourly precip amts on the grid

p12 12-hourly precip amts on the grid

quad9r Missing value (9999)

sixprc Array of six hourly precip amounts

startjul Starting julian hour of one hour loop

stncnt Number of observations for this time period

twfprc Array of 24 hourly precip amounts

use_twelve Logical flag to use the 12-hourly precip amts or the 6 hourly amts

The routines invoked are:

julhr_date (5.4.23)

converts julian hour to a date format

agrmetpcpobsfilename (29.1.66)

generates the name of the precip observations file

getpwe (29.1.68)

get present weather estimate

storeobs (29.1.70)

store observations

setpathpcpobs (29.1.67)

set the path to write the observations analysis

processobs (29.1.71)

process observations

29.1.66 agrmetpcpobsfilename (Source File: getpcpobs.F90)

INTERFACE:

```
subroutine agrmetpcpobsfilename(name,dir,hemi,yr,mo,da,hr)
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: hemi
integer, intent(in) :: yr,mo,da,hr
character*100      :: name
character*50       :: dir
```

DESCRIPTION:

This routines generates the name of the precip observations file by appending the hemisphere and timestamps to the root directory.

The arguments are:

hemi index of the hemisphere (1-NH, 2-SH)
dir full path to the directory containing the data
name created filename
yr 4 digit year
mo integer value of month (1-12)
da day of the month
hr hour of the day

29.1.67 setpathpcpobs (Source File: getpcpobs.F90)

INTERFACE:

```
subroutine setpathpcpobs(name,dir,yr,mo,da,hr)
implicit none
```

ARGUMENTS:

```
character*100 :: name
character*50  :: dir
integer, intent(in) :: yr,mo,da,hr
```

DESCRIPTION:

This routines generates the name of the path to which processed precip observations are to be written.
The arguments are:

dir full path to the directory containing the data

name created filepath
yr 4 digit year
mo integer value of month (1-12)
da day of the month
hr hour of the day

29.1.68 getpwe (Source File: getpwe.F90)

REVISION HISTORY:

```

20 oct 89 initial version.....mr moore/sddc(agromet)
20 may 91 added check for octal 7 value indicating missing rpt.
           updt prolog.....mr moore/sddc(agromet)
15 jun 96 added new variables (arguemnts) newd and olld to be
           passed thru to setprc routine. updated prolog and
           brought up to stds.....mr moore/sysm(agromet)
03 apr 97 added the variables wmonum and ztime to the called
           routine setprc so that they may be used by the
           routine.....ssgt miller/sysm
04 dec 97 revised routine to call rainbo for bogus values instead
           of retrieving the values from database. brought up to
           standards and updated prolog..capt andrus/dnxm(agromet)
 7 oct 99 ported to ibm sp-2, updated prolog, incorporated
           FORTRAN 90 features.....capt hidalgo/agrmet
21 feb 01 removed call to routine setprc (which placed estimates
           on the grid) and moved its logic within this routine.
           moved observation loop and data checks to this
           routine because routines onhr1, offhr1 and offhr2
           were replaced with new precip decoder.....mr gayno/dnxm

```

INTERFACE:

```

subroutine getpwe (nsize, isize, bsn, hemi, ilat, ilon,&
                   month, pastwx, preswx, olld, &
                   prcpwe, imax, jmax)

implicit none

```

ARGUMENTS:

integer,	intent(in)	:: isize
integer,	intent(in)	:: bsn(isize)
integer,	intent(in)	:: hemi
integer,	intent(in)	:: ilat(isize)
integer,	intent(in)	:: ilon(isize)
integer,	intent(in)	:: imax
integer,	intent(in)	:: jmax
integer,	intent(in)	:: month
integer,	intent(in)	:: nsize
integer,	intent(in)	:: pastwx(isize)
integer,	intent(in)	:: preswx(isize)

```
real,           intent(inout) :: olld(imax,jmax)
real,           intent(inout) :: prcpwe(imax,jmax)
```

DESCRIPTION:

to determine precipitation amounts using past and present weather codes.

Method

- loop through all observations for this time period.
- check for valid lat/lon, wmo station id, present/past weather code.
- if observation is valid, call routine makpwe to calculate an estimate.
- place this estimate at the nearest grid point. if there is more than one estimate within one grid box, use the nearest one.

The arguments and variables are:

a,b,c,d holds i/j coordinates of estimate/grid point in function sumsqr

bsn wmo block station number

dumhemi hemisphere flag passed to utility routine lltops

hemi hemisphere flag (1 - northern, 2 - southern)

ilat observation latitude

ilon observation longitude

imax grid dimension, i-direction

irecord loop index

isize maximum size of observation arrays

jmax grid dimension, j-direction

month current month

newd distance (in grid lengths) between estimate and the nearest grid point

nsize number of observations in the arrays

oldd holds distances between estimates and the nearest grid point

pastwx past weather code

preswx present weather code

ri/rj i/j coordinate of an estimate in grid point space

rlat/rlon lat/lon of an estimate

sumsqr function to calculate the distance between an estimate and the nearest grid point

wxest precipitation estimate passed back from makpwe

The routines invoked are:

makpwe (29.1.69)

create a present weather estimate

lltops (7.0.24)

converts a lat lon values to a corresponding point on the AGRMET grid

29.1.69 makpwe (Source File: makpwe.F90)

REVISION HISTORY:

```
04 dec 97 initial version.....mr moore, capt andrus/dnxm(agromet)
 7 oct 99 ported to ibm sp-2, updated prolog, incorporated
          FORTRAN 90 features.....capt hidalgo/agrmet
```

INTERFACE:

```
subroutine makpwe( preswx, pastwx, lat, month, wmonum, pwprc )
```

DESCRIPTION:

to create an estimate of precipitation from the present and past weather codes

Method

1. calculate a 1st guess precip rate by combining the array values for the reported present and past wx.
2. determine a wmo region adjustment.
3. determine a gross latitude adjustment.
4. calculate a seasonal latitude adjustment.
5. adjust the 1st guess precip rate with the three adjustment factors and convert the result to mm/hr.

The arguments and variables are:

blkadj array of wmo block adjustments

diff obs site's displacement from latitude of max precip

fguess first guess total present weather estimate value

gross gross latitudinal adjustment

lat latitude of the observation site

maxlat monthly varying latitude of max precipitation

month current month of the observation

pastfg selected first guess past wx estimate value

pastamt array of possible first guess past wx values

pastwx past weather code reported in the observation

presfg selected first guess present wx value

presamt array of possible first guess present wx values

preswx present weather code reported in the observation

pwprc final precipitation rate (mm/hr)

season calculated seasonal latitude adjustment

wmoadj selected wmo block adjustment

wmoblk wmo region number for the report

wmonum wmo block station number

29.1.70 storeobs (Source File: storeobs.F90)

REVISION HISTORY:

```
21 feb 01 initial version.....mr gayno/dnxm
25 feb 02 store misc amounts for russian obs.....mr gayno/dnxm
```

INTERFACE:

```
subroutine storeobs(nsize, isize, obs, ilat, ilon, &
    mscprc, sixprc, twfprc, bsn, &
    duration, stncnt)

implicit none

integer, intent(in) :: isize
integer, intent(in) :: bsn(isize)
integer, intent(in) :: duration(isize)
integer, intent(in) :: ilat(isize)
integer, intent(in) :: ilon(isize)
integer, intent(in) :: mscprc(isize)
integer, intent(in) :: nsize
integer, intent(in) :: sixprc(isize)
integer, intent(inout) :: stncnt
integer, intent(in) :: twfprc(isize)
```

DESCRIPTION:

performs some preprocessing on the raw observations and stores valid data in a storage array. **Method**
- ensure observation has a valid latitude, longitude, wmo number and at least one valid precip amount.
- if there is a valid 24 hourly amount...
- if amount is zero, then we know the 12 and 6 hourly amounts must be zero as well.
- if amount is less than 1 mm, as indicated by a flag of -91 thru -98, set amount to 1 mm.
- store amount in 24 hourly part of obs data structure. - if there is a valid 12 hourly amount...
- if amount is zero, then we know the 6 hourly amount must be zero as well.
- if amount is less than 1 mm, as indicated by a flag of -91 thru -98, set amount to 1 mm.
- store amount in 12 hourly part of obs data structure - if there is a valid 6 hourly amount...
- if amount is less than 1 mm, as indicated by a flag of -91 thru -98, set amount to 1 mm.
- store amount in 6 hourly part of obs data structure - if the observation is from india...
- all precip amounts are reported from 03z. store this amount in the 12 hourly amount place holder even though it is not a 12 hourly amount. this special case is handled in subsequent subroutines.
- if the observation is from russia (or any part of the former soviet union) store the misc amount. russia reports every 12 hours but does not use a valid duration flag. therefore, the misc amount will be used as a 12 hourly amount in later routines.

The arguments and variables are:

bsn wmo block station number array

duration valid period of precipitation

ilat array of observation latitudes

ilon array of observation longitudes

irecord loop index

isize maximum size of observation arrays
mscprc miscellaneous precip amounts
nsize number of obs returned from database
obs array of processed observations
 amt6 six hourly precip amount
 amt12 twelve hourly precip amount
 amt24 twenty-four hourly precip amount
 lat latitude
 lon longitude
 wmonum wmo block station number

rlat temporary holding variable for a latitude
rlon temporary holding variable for a longitude
sixprc array of six hourly precip amounts
stncnt number of observations with a precip report
temp6 temporary holding variable for a 6 hourly amount
temp12 temporary holding variable for a 12 hourly amount
temp24 temporary holding variable for a 24 hourly amount
tempmsc temporary holding variable for a misc amount
twfprc array of 24 hourly precip amounts

29.1.71 processobs (Source File: processobs.F90)

REVISION HISTORY:

```

21 Feb 01 Initial version.....Mr Gayno/DNXM
25 Feb 02 Modified to use misc array over Russia....Mr Gayno/DNXM
07 Jan 03 Expanded use of Russian obs at 00/12 utc. Use all
          available obs with block station numbers between
          20000 and 38000.....Mr Gayno/DNXM
17 Mar 04 Removed duplicate "/" from filenames...Mr Lewiston/DNXM

```

INTERFACE:

```

subroutine processobs(obs, isize, stncnt, hemi, julhr, &
    imax, jmax, p6, pathpcp, p12,&
    use_twelve, quad9r, alert_number)

```

USES:

```

use lis_logmod, only : lis_alert
use spmdMod
use lis_logmod, only  : logunit
implicit none

```

ARGUMENTS:

```
character*100, intent(in)      :: pathpcp
integer,  intent(in)          :: imax
integer,  intent(in)          :: jmax
real,    intent(inout)        :: p6(imax,jmax)
real,    intent(inout)        :: p12(imax,jmax)
real,    intent(in)          :: quad9r
integer,  intent(inout)        :: alert_number
integer,  intent(in)          :: hemi
integer,  intent(in)          :: isize
integer,  intent(in)          :: julhr
integer,  intent(in)          :: stncnt
logical, intent(in)          :: use_twelve
```

DESCRIPTION:

Processes rain gauge data then maps it to the model grid. The idea behind the processing is to make as many 6 and 12 hourly rain gauge amounts as possible using the information that each station provides.

Method

- Sort the array of observations by WMO number.
- Filter out duplicate obs (those with the same WMO number).
- Read in observations from 6 hours ago (if available).
- Read in observations from 12 hours ago (if available).
- If the current 12 hourly amount is missing, try to calculate it from current 6 hourly amts and 6 hourly amts from 6 six hours ago.
- If the above is not possible, then try to create the current 12 hourly amt from a current 24 hourly amt and a 12 hourly amt from 12 hours ago.
- If the current 6 hourly amt is missing, then try to calculate it using a current 12 hourly amt and a 6 hourly amt from six hours ago.
- If the station is from India/Bangladesh, then calculate current 12 and 6 hourly observations according to their own renegade reporting practices.
- Write the final array of current observations to an ascii file for use the next time precip runs.
- If this is an 06 or 18 utc run, put the current 6 hourly amts on the model grid.
- If this is an 00 or 12 utc run, put the current 12 hourly amts on the model grid.
- Data over S America and Russia require special handling.

The arguments are variables are:

a,b,c,d Holds i/j coordinates of estimate/grid point in function sumsqr

alert_number Number of alert messages

chemi Character representation of hemisphere

count Position of a filtered observation in the obs_cur array

count6 Number of observations from 6 hours ago

count6obs Number of 6 hourly rain gauge observations at the current time

count12 Number of observations from 12 hours ago

count12obs Number of 12 hourly rain gauge observations at the current time

count_dup Number of duplicate observations

date10 Date/time group (yyyymmddhh) of current time

date10_min6 Date/time group of current time minus six hours
date10_min12 Date/time group of current time minus twelve hours
dumhemi Dummy variable to hold hemisphere flag
filename Name, including path, of the current observation file
filename_min6 Name, including path, of the observation file from six hours ago
filename_min12 Name, including path, of the observation file from twelve hours ago
fltrcnt Number of observations after duplicates are filtered out
hemi Hemisphere flag (1 - N, 2 - S)
i Loop index
index6 Array index of an observation (from 6 hours ago)
index12 Array index of an observation (from 12 hours ago)
imax Grid dimension - E/W direction
iofunc I/O function - read or write. Used for diagnostic prints
isize Maximum array size of observation arrays
istat i/o status
jmax Grid dimension - n/s direction
julhr Julian hour of the current time
julhr_min6 Julian hour of the current time minus 6 hrs
julhr_min12 Julian hour of the current time minus 12 hrs
locsmall Position of the smallest WMO number in the unsorted observation array
message Alert message
newd Distance in grid lengths between an observation and the nearest model grid point
obs Array of unsorted observations (by WMO number)
obs_cur Array of sorted observations (by WMO number) for the current time
obs_6 Array of observations from 6 hours ago
obs_12 Array of observations from 12 hours ago
oldd Holds distances between an observation and the nearest grid point
pathpcp Directory path of precip data
p6 Six hourly precip amts on the model grid
p12 Twelve hourly precip amts on the model grid
quad9r The missing indicator (9999.)

rain_obs Abstract data type used to hold rain gauge obs. contains the following fields:

amt6 - six hourly precip amount
amt12 - twelve hourly precip amount
amt24 - twenty-four hourly precip amount
amtmisc - miscellaneous precip amount
lat - latitude
lon - longitude
wmonum - WMO block station number

ri I coordinate of an observation on the model grid

rj J coordinate of an observation on the model grid

sixyes Logical flag - does obs data exist from six hours ago

smalloc Position, counting from 1, of the smallest WMO number in the unsorted observation array between i and stncnt

smallwmo Smallest WMO number in the unsorted observation array

stncnt Number of observations with a precip report

sumsqr Function to calculate the distance between an Estimate and the nearest grid point

temp6 Temporary holding variable for 6hrly precip amt

temp12 Temporary holding variable for 12hrly precip amt

temp24 Temporary holding variable for 24hrly precip amt

templat Temporary holding variable for latitude

templon Temporary holding variable for longitude

tempmsc Temporary holding variable for misc precip amt

twelveyes Logical flag - does obs data exist from twelve hours ago

use_twelve Logical flag to put 12 hourly precip amts on the model grid (set to true for 00 and 12 utc runs).
when false (at 06 and 18 utc), put 6 hourly precip amts on the model grid

Remarks

This routine has special logic to handle some countries' renegade reporting practices (for example, India). Therefore, the maintainer should remain alert to changes in reporting practices and modify the code as necessary.

The routines invoked are:

julhr_date10 (29.1.90)
converts julian hour to a 10 character date string

lltops (7.0.24)
converts lat lon values to points on the AGRMET grid

pcpobs_search (29.1.72)
finds a specific observation in an array of observations sorted by WMO number.

lis_alert (5.24.3)
prints out an alert message

29.1.72 pcpobs_search (Source File: pcpobs_search.F90)

REVISION HISTORY:

21 feb 01 initial version.....mr gayno/dnxm

INTERFACE:

```
subroutine pcpobs_search(wmonum, isize, obs, index)
```

```
    implicit none
```

ARGUMENTS:

```
    integer, intent(out)      :: index  
    integer, intent(in)       :: isize  
    integer, intent(in)       :: wmonum
```

DESCRIPTION:

finds a specific observation in an array of observations sorted by wmo number.

Method

- search for an observation using a binary search.
- if found, pass back its array index.
- if not found, pass back an array index of -9999 (missing).

The arguments are variables are:

first array index of the beginning of the search region

found logical flag - true/false - observation found

index array index of the observation we were searching for. if not found, set to -9999

isize number of elements in the observation array

last array index of the end of the search region

middle array index of the middle of the search region

obs array of observations sorted by wmo number

amt6 six hourly precip amount

amt12 twelve hourly precip amount

amt24 twenty-four hourly precip amount

lat latitude

lon longitude

wmonum wmo block station number

wmonum wmo block station number we are searching for

29.1.73 makest (Source File: makest.F90)

REVISION HISTORY:

1 jul 89 initial version.....mr moore/sddc(agromet)
 11 feb 91 added 'land' and 'estwt' to arg list of getcld. added
 'prob' to makest and calest arg list. eliminated call
 to setcoe by adding call to bndslc and adding some
 former setcoe code up in makest btwn calls to bndslc
 and calest. reorganized prolog method discussion. re-
 sized pcoef and pcout arrays to reflect new latbnd
 and cldtyp array size. isolated debug stmt. declared
 misc loop variables.....mr moore/sddc(agromet)
 04 may 91 chgd init of 'ra' and 'e' arrays to 9999.0 so rainrates
 of 0.0 can be used. chg logic in setting rpcoef. added
 'thres' to arg lists so as to pass it down thru getcld
 to lodcld to eliminate multiple calls to func 'int'.
 updtd prolog.....mr moore/sddc(agromet)
 3 mar 92 made chgs necessary to include cldamt as an index of
 pcoef and pcout arrays. fixed error in process to pick
 rpcoef. movd read/write of pcoef up to driver and de-
 leted pcout array. increased the # of cld types used
 mr moore/sys(agromet)
 1 jun 92 created cac to make cld amt index of pcoef adjustable
 fm control file. added use of rh class and stability
 class as indices to pcoef. expanded pcoef array and
 reduced prob array. added use of pwgt.....
 mr moore/sys(agromet)
 1 feb 94 deleted stability class, rh class, cldtyp and latbnd
 as pcoef indices. added cldtop ht class and tropopause
 ht class as pcoef indices. added calls to geoslc (in
 place of bndslc) and zrobog (to zero-fill around bogus
 values of zero. omitted write stmts that produced
 smiedr status print. expanded cldamt array to account
 for 3-hrly times. updtd prolog.....
 mr moore, capt bertone, sys(agromet)
 25 apr 94 added 'files accessed' section as 'id'ed during the
 94-01 qaa..... capt bertone sysm(agromet)
 15 jun 96 eliminated loading of previous precip estimates in
 back end of 'e' array. reduced array 'e' size from
 (64,64,12) to (64,64,4). eliminated writing of 3-hrly
 estd precip amts to file, a function now handled in
 phsrel routine. moved calc of pprob from calest to
 to this routine. updtd prolog and brought up to stds.
 mr moore sysm(agromet)
 07 mar 97 added subroutine crest to create an estimate based on
 climatological precip histories, which replaced the
 precip coefficient and tropopause height class estimate
 routines. altered the arrangement of the code to run
 more efficiently. brought code up to current afgwc and
 sysm standards.....capt andrus, ssgt mccormick/sysm
 24 mar 97 added the 'source' array which is argued through this
 routine into the 'calest' routine.....ssgt miller/sysm
 19 may 97 redimensioned source array to write each 3-hour sourced
 estimate into its own file.....ssgt miller/sysm
 04 dec 97 changed crest routine to only produce a rtnehp estimate
 and renamed rtnest. the rtnest routine also now outputs
 rtnehp precip estimates. modified makest routine to call
 cliest for a pure climo based estimate and smiest for

an ssmi based estimate. added getcli routine to retrieve monthly climo amounts and interpolate monthly values to the current day. modified calest to identify all source type based upon a pure hierarchy. file assign also now assigns the additional and rtneph data files. updated prolog and brought up to standards.....
capt andrus/dnxm(agromet)
 2 sep 99 added option for geoswch to be 0, 1, or 2. this allows the rank values computed by geo_precip to either be used (geoswch=2) or not used (geoswch=1) in the final calculation of the precip estimate.....capt hidalgo/dnxm
 07 oct 99 ported to ibm sp-2, updated prolog, incorporated FORTRAN 90 features.....capt hidalgo/agrmet
 09 may 00 modified list of arguments in call to smiest.....
capt hidalgo/agrmet
 01 jun 00 removed estwt from program. estwt was used to control the way different estimates are ordered for use in the calest sub-routine. changed grnk to integer, added grnk to calest call.....ssgt campbell/agrmet
 21 feb 01 modified loop and changed some variable names to reflect change to 6-hrly cycles.....mr gayno/dnxm
 10 jun 02 modified for incorporation of cdfs2 data to replace rtneph.....mr gayno/dnxm
 29Jul2005 Sujay Kumar, Initial Code

INTERFACE:

```
subroutine makest(n, hemi,j6hr,estpcp,source,cdfs2est,prcpwe)
```

USES:

```

use lisdrv_module, only : lis
use agrmetdomain_module, only : agrmet_struc

implicit none

integer, intent(in) :: n
integer, intent(in) :: hemi
integer, intent(in) :: j6hr
real, intent(out) :: estpcp(agrmet_struc(n)%imax,agrmet_struc(n)%jmax,2)
integer, intent(out) :: source(agrmet_struc(n)%imax,agrmet_struc(n)%jmax,2)
real, intent(out) :: cdfs2est(agrmet_struc(n)%imax,agrmet_struc(n)%jmax,2)
real :: prcpwe(agrmet_struc(n)%imax, agrmet_struc(n)%jmax,2)

```

DESCRIPTION:

to make a consolidated 'final' precipitation amount estimate based upon several possible precipitation estimates.

Method

- initialize output arrays.
- retrieve and interpolate any needed climatological data.
- using cliest, make pure climo precip estimate if desired.
- loop back through the 6-hour cycle at 3-hour increments
- initialize local precip estimate arrays.
- if specified use ssmi data then generate the ssmi estimates using smiest.

- if specified use climo precip data and cdfs2 cloud data to generate a cdfs2 based precip estimate in routine `cdfs2.est`.
- using specified hierarchy and data availability determine an estimate and identify the source of the estimate for every grid point using subroutine `calest`.

The arguments and variables are:

alert_number alert_number
cdfs2est cdfs2 cloud based precip estimate (mm/3hr)
cdfs2int cdfs2 time interval to look for cloud amount
cdfs2swch cdfs2-based estimate use switch
cest pure climatological based precip estimate (mm/3hr)
estpcp final estimated precip amounts (mm/3hr)
gest geo-precip estimated precip values
grnk geo-precip rank values for goodness of estimates 1 = better than ssmi est. 2 = better than present wx based est. 3 = better than cdfs2 est. 4 = default value of geo-precip est. 5 = worse than climatological est.
hemi hemisphere (1=north, 2=south)
j6hr julian hour of beginning of 12-hour period
j3hr loop index, 3-hour julian hour
k loop counter (1 through 4)
prcpwe bogus precip rates (mm/3hr)
ra ssmi precip estimate
source array that contains source-of-the-estimate information with specified source flags

The routines invoked are:

read_pcpclimodata (29.1.11)
 read precip climatology files
cliest (29.1.74)
 create purely climatologically based precip estimate
smiest (29.1.75)
 create an SSM/I based estimate
cdfs2_est (29.1.77)
 CDFS2-based estimate
geoest (29.1.79)
 GEOPRECIP-based estimtate
calest (29.1.82)
 calculate a final estimate

29.1.74 cliest (Source File: cliest.F90)

REVISION HISTORY:

```
04 dec 97 initial version.....capt andrus/dnxm(agromet)
31 mar 99 changed array and loop sizes to hemisphere size.
.....mr moore/dnxm
7 oct 99 ported to ibm sp-2, updated prolog, incorporated
FORTRAN 90 features.....capt hidalgo/agrmet
29Jul2005 Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine cliest( hemi, cest, cliprc, clmult, land, quad9r, &
imax, jmax )

implicit none
```

ARGUMENTS:

```
integer :: hemi
integer, intent(in) :: imax
integer, intent(in) :: jmax
real :: cest(imax,jmax)
real :: cliprc(imax,jmax)
integer :: land(imax,jmax,2)
real :: clmult
real :: quad9r
```

DESCRIPTION:

to create a precipitation estimate based on climatological precip amounts.

Method

- for all land points
- check validity of 3-hour climo precip
- if valid, replace initialized climo estimate with climo precip amount multiplied by the mult factor read in from the control file.
- if not valid, print out a message to that effect.

The arguments and variables are:

cest pure climatological precipitation estimate (mm/3hr)
cliprc 3-hour climo precip used for estimate
clmult alternate monthly weighting factor
hemi hemisphere (1=nh, 2=sh)
i loop counter
imax number of gridpoints in east/west direction
j loop counter
jmax number of gridpoints in north/south direction
land points of major land masses
quad9r value of 9999.0 for initializing arrays

29.1.75 smiest (Source File: smiest.F90)

REVISION HISTORY:

```
04 dec 97 initial version.....capt andrus/dnxm(agromet)
09 may 00 modified code to read in values from ported version
          of smiedr.....capt hidalgo/dnxm
29Jul2005 Sujay Kumar, Initial Code in LIS
```

INTERFACE:

```
subroutine smiest( n, hemi, j3hr, quad9r, ra, &
                  razero, imax, jmax, alert_number)
```

USES:

```
use agrmetdomain_module, only : agrmet_struct
use lis_fileIOMod, only : putget
use listime_mgr, only : julhr_date
use lis_logmod, only : lis_alert, logunit

implicit none
```

ARGUMENTS:

integer,	intent(in)	:: n
integer,	intent(in)	:: hemi
integer,	intent(in)	:: j3hr
integer,	intent(in)	:: imax
integer,	intent(in)	:: jmax
real,	intent(out)	:: ra(imax,jmax)
integer,	intent(in)	:: razero
real,	intent(in)	:: quad9r
integer		:: alert_number

DESCRIPTION:

to make an ssmi based estimate array.

Method

- retrieve rain rates for the current hemisphere from file.
- the files retrieved will be 3hrly amounts (mm)
- if the razero flag equals 1, reset the ssmi zeros to quad9r

The arguments and variables are:

n index of nest

alert_number alert message number

exists a logical that indicates whether or not a file exists

hemi hemisphere (1=nh, 2=sh)

imax maximum number of gridpoints in x-direction

ifil input ssmi data file name

jmax maximum number of gridpoints in y-direction

message array of alert messages
quad9r parameter for 9999.0 value
ra ssmi based estimated precip amount (mm/3hr)
razero flag to use zero ssmi amounts (1=use zeros)
use_zeros logical variable that is true if ssmi zeros are to be used
The routines invoked are:
julhr_date (5.4.23)
 converts julian hour to a date format
agrmet_ssmiprec_filename (29.1.76)
 generates the SSM/I filename
putget (5.23.1)
 read the SSM/I data
lis_alert (5.24.3)
 print out an alert message in case of missing data

29.1.76 agrmet_ssmiprec_filename (Source File: smiest.F90)

INTERFACE:

```
subroutine agrmet_ssmiprec_filename(name,dir,hemi,yr,mo,da,hr)
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: hemi
integer, intent(in) :: yr,mo,da,hr
character*100      :: name
character*50        :: dir
```

DESCRIPTION:

This routines generates the name of the SSM/I file to be read, with the appropriate hemisphere and time stamps.

The arguments are:

hemi index of the hemisphere (1-NH, 2-SH)
dir full path to the directory containing the data
name created filename
yr 4 digit year
mo integer value of month (1-12)
da day of the month
hr hour of the day

29.1.77 cdfs2_est (Source File: cdfs2_est.F90)

REVISION HISTORY:

```
07 mar 97 initial version.....ssgt mccormick/sysm
24 mar 97 added the 'source' array which gives and sets each
           estimate type with a specified integer source flag;
           determining the values with the highest estimate
           information.....ssgt miller/sysm
19 may 97 redimensioned source array to write each 3-hour sourced
           estimate into its own file.....ssgt miller/sysm
04 dec 97 changed name of routine from crest to rtnest to more
           correctly reflect functionality. modified to generate
           rtnehp based estimates for all points and not climo
           estimates. also removed determination of a source from
           this routine. updated prolog and brought up to
           standards.....capt andrus/dnxm(agromet)
31 mar 99 changed box looping to hemisphere looping and added new
           file retrieval convention for rtnehp data. also added
           abort message to cover division by zero possibility ...
           .....mr moore/dnxm
7 oct 99 ported to ibm sp-2, updated prolog, incorporated
           FORTRAN 90 features.....capt hidalgo/agrmet
31 mar 00 changed variable "ifil" to character*120 to be
           consistent with routine copen. added error checks
           after calls to copen and gribread.....mr gayno/dnxm
11 may 00 changed variable rtntim from integer to real. this
           fixed a problem degribbing the data.....
           .....capt hidalgo, mr gayno/dnxm
21 feb 01 reduced third dimension (number of three hourly time
           periods) of variable rest from 4 to 2 as module precip
           runs in 6 hourly cycles now.....mr gayno/dnxm
10 jun 02 modified to use cdfs2 data instead of rtnehp.....
           .....mr gayno/dnxm
3 nov 2005 Sujay Kumar, incorporated into LIS
```

INTERFACE:

```
subroutine cdfs2_est( n,hemi, k, land,cliprc, clippd,&
    clirtn, mnpr, mxpr, mnpd, mxpd, &
    cldth, mdmv, mdpe, ovpe, &
    cdfs2int, cdfs2est, imax, jmax,&
    alert_number, j3hr )
```

USES:

```
use agrmetdomain_module, only : agrmet_struct
use listime_mgr, only : julhr_date
use lis_logmod, only : lis_alert, lis_abort, logunit
implicit none
```

ARGUMENTS:

```
integer,      intent(in)      :: n
integer,      intent(in)      :: hemi
```

```

integer,      intent(in)      :: imax
integer,      intent(in)      :: jmax
integer,      intent(in)      :: j3hr
integer,      intent(inout)   :: alert_number
integer,      intent(in)      :: cdfs2int
integer,      intent(in)      :: k
integer,      intent(in)      :: land(imax,jmax,2)
real,         intent(out)    :: cdfs2est(imax,jmax,2)
real,         intent(in)      :: cldth
real,         intent(in)      :: clippd(imax,jmax,2)
real,         intent(in)      :: cliprc(imax,jmax,2)
real,         intent(in)      :: clirtn(imax,jmax,2)
real,         intent(in)      :: mdmv
real,         intent(in)      :: mdpe
real,         intent(in)      :: mnpd
real,         intent(in)      :: mnpr
real,         intent(in)      :: mxpd
real,         intent(in)      :: mxpr
real,         intent(in)      :: ovpe

```

DESCRIPTION:

to create a precipitation estimate based on climatological precip amount, climatological precip per precip day amounts, climatological rtnehp cloud amounts and actual cdfs2 total cloud amounts.

Method

- retrieve current cloud data and pixel times from file.
- if current data is missing or on a bad read, search back a maximum of two hours for a complete set of data. if this search fails, then don't create an estimate for this time period.
- for each land point in the hemisphere:
- if valid time for this point is within the window
- if current cdfs2 total cloud data is valid and is at least equal to the minimum cdfs2 total cloud amount threshold.
- if the climatological precip amount for this grid point meets or exceeds the minimum required climo precip amount for a cdfs2 based precip estimate.
- if current cdfs2 cloud amount meets or exceeds climo rtnehp cloud amount, but is less than 100 percent.
- determine climo precip mult factor.
- determine precip-per-precip day mult factor based upon cloud amount.
- calculate estimate by multiplying precip-per-precip day by the climo precip mult factor, the precip-per-precip day factor, and the monthly rthneph factor.
- else estimate zero precip
- else estimate zero precip
- else estimate zero precip
- else do not estimate precip

The arguments and variables are:

n index of the nest

alert_number alert number

cdfs2est cdfs2 cloud amount based precip est (mm/3hr)

cdfs2int cdfs2 time interval to look for cloud amount

cldamt current cdfs2 cloud amount

cldth cloud threshold to generate a cdfs2 based precipitation estimate

cldtim cdfs2 cloud pixel time on the agrmet grid (time of newest data at gridpoint)

clifac climatological multiplication factor

clippd climatological precip-per-precip day amount (mm/3hr) interpolated to the current day

cliprc climatological monthly precip amount (mm/3hr) for interpolated to the current day

clirtn climatological rtneph percent cloud cover interpolated to the current day

date10 3-hrly 10-digit date time group (YYYYMMDDHH)

error_flag logical flag for good/bad cdf2 data ingest

hemi hemisphere (1=north, 2=south)

i i-coord loop counter

icdfs2 i-dimension of cdfs2 grid

ifl dummy character array for input file name

imax number of gridpoints in east/west direction

istat i/o status

j j-coord loop counter

jcdfs2 j-dimension of cdfs2 grid

jmax number of gridpoints in north/south direction

j3hr julian hour of end of precip accumulation period

k 3-hrly loop counter

land points of major land masses

maxth max actual cdfs2 cloud cover threshold for max precip estimate

mdmv median cloud cover percentage to move to for the cdfs2 based precipitation estimate

mdpe percent to move to median cloud cover percentage to move to for the cdfs2 based precipitation estimate

message message array for abort

minth min actual cloud cover threshold for min precip estimate

mnpd minimum precip-per-precip day multiplier used to generate a non-zero cdfs2 total cloud based precip estimate

mxpd maximum precip-per-precip day multiplier used to generate a non-zero cdfs2 total cloud based precip estimate

mnpr minimum 3-hour climo precip value required to generate a non-zero cdfs2 total cloud based precip estimate

mxpr max 3-hour climo precip value at which the maximum precip-per-precip day multiplier is used to generate a cdfs2 total cloud based precip estimate

ovpe overcast percentage to move to for cdfs2 based precipitation estimate

ppddif difference between mxrppd and mnrrpd
ppdfac precip-per-precip day mult factor based upon cdfs2 cloud amount
time loop index for reading in cdfs2 data
times cdfs2 pixel times
totcld array of cdfs2 total cloud amounts on the agrmet grid

The routines invoked are:

- julhr_date** (5.4.23)
converts julian hour to a date format
- agrmet_cdfs_totalcld_filename** (29.1.78)
generates the filename for CDFS2 total cloud data
- julhr_date10** (29.1.90)
converts julian hour to a 10 character date string
- lis.alert** (5.24.3)
prints a alert message
- lis.abort** (5.24.2)
aborts if a fatal error occurs
- agrmet_cdfs_pixltime_filename** (29.1.22)
generates the filename to read CDFS2 pixel times

29.1.78 agrmet_cdfs_totalcld_filename (Source File: cdfs2.est.F90)

INTERFACE:

```
subroutine agrmet_cdfs_totalcld_filename(name,dir,hemi,yr,mo,da,hr)
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: hemi
integer, intent(in) :: yr,mo,da,hr
character*120      :: name
character*50        :: dir
```

DESCRIPTION:

This routines generates the name of the CDFS2 total cloud data by appending the hemisphere and timestamps to the root directory.

The arguments are:

- hemi** index of the hemisphere (1-NH, 2-SH)
- dir** full path to the directory containing the data
- name** created filename
- yr** 4 digit year

mo integer value of month (1-12)

da day of the month

hr hour of the day

29.1.79 geoest (Source File: geoest.F90)

REVISION HISTORY:

```
31 mar 99 initial version ..... mr moore/dnxm
 2 sep 99 added option for geoswch to be 0, 1, or 2.  this allows
        the rank values computed by geo_precip to either be
        used (geoswch=2) or not used (geoswch=1) in the final
        calculation of the precip estimate.....capt hidalgo/dnxm
 8 oct 99 ported to ibm sp-2, updated prolog, incorporated
        FORTRAN 90 features.....capt hidalgo/agrmet
 01 jun 00 changed grnk to integer, changed geornk to integer,
        initialized gdgeornk to false. Included error checking
        for grnk values when used. Added alert message if no
        grnk file is found when grnks are being used. Removed
        variable k from passed variables and declarations, could
        not find anywhere that variable was used.....
        .....ssgt campbell/dnxm
 21 feb 01 reformatted the diagnostic prints.....mr gayno/dnxm
 10 jun 02 changed all references to rtneph to cdfs2..mr gayno/dnxm
 3 nov 2005 Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine geoest( n, hemi, j3hr, land, gest, grnk, quad9r, &
                   imax, jmax, geoswch,&
                   alert_number )
```

USES:

```
use agrmetdomain_module, only : agrmet_struct
use lis_fileIOMod, only      : putget
use listime_mgr, only        : julhr_date
use lis_logmod, only         : lis_alert, logunit

implicit none
```

ARGUMENTS:

integer, intent(in)	:: imax
integer, intent(in)	:: jmax
integer, intent(in)	:: n
integer, intent(in)	:: j3hr
integer, intent(inout)	:: alert_number
integer, intent(in)	:: geoswch
integer, intent(inout)	:: grnk(imax,jmax)
integer, intent(in)	:: hemi
integer, intent(in)	:: land(imax,jmax,2)
real, intent(out)	:: gest(imax,jmax)
real, intent(in)	:: quad9r

DESCRIPTION:

to retrieve a precipitation estimate based on the geo-precip technique

Method

- read geo-precip file for this time/hemisphere
- if file does not exist, return to calling routine. Geo-precip date will not be used. Send alert message.
- if geoswch equals 2, read in ssmi values, if file exists.
- loop over hemisphere
- replace geo-precip "missing" values w/9999.0
- if value is valid and over land, place precip amount in output array gest.
- if geoswch equals 2 and rank value is valid, place in output array grnk.
- if rank value invalid, set to default value of 4

The arguments and variables are:

n index of the nest

alert_number alert number

exists a logical that indicates whether or not a file exists

gdgeornk logical indicating if the rank values are usable

geoprc temp array used to read in geo precip values

goornk temp array used to read in rank values

geoswch geo-precip estimate switch

- 0 = don't,
- 1 = process w/o grnk
- 2 = process with grnk

gest final geo-precip-based precip estimates (mm/3hr)

grnk final geo-precip rank values

- 1 = better than ssmi est.
- 2 = better than present wx based est.
- 3 = better than cdfs2 est.
- 4 = default value of geo-precip est.
- 5 = worse than climatological est.

hemi hemisphere (1=nh, 2=sh)

i loop counter

ifil dummy character array for input file name

imax number of gridpoints in east/west direction

j loop counter

jmax number of gridpoints in north/south direction

land land/sea mask (0 = water, 1 = land)

message message array for alert

quad9r value of 9999.0 for initializing arrays

The routines invoked are:

julhr_date (5.4.23)

converts julian hour to a date format

agrmet_geoprec_filename (29.1.80)

agrmet_geornk_filename (29.1.81)

lis_alert (5.24.3)

prints a alert message

putget (5.23.1)

retrieves data from the specified file

29.1.80 agrmet_geoprec_filename (Source File: geoest.F90)

INTERFACE:

```
subroutine agrmet_geoprec_filename(name,dir,hemi,yr,mo,da,hr)
  implicit none
```

USES:

```
integer, intent(in) :: hemi
integer, intent(in) :: yr,mo,da,hr
character*100      :: name
character*50       :: dir
```

DESCRIPTION:

This routines generates the name of the GEO precip file by appending the hemisphere and timestamps to the root directory.

The arguments are:

hemi index of the hemisphere (1-NH, 2-SH)

dir full path to the directory containing the data

name created filename

yr 4 digit year

mo integer value of month (1-12)

da day of the month

hr hour of the day

29.1.81 agrmet_georank_filename (Source File: geoest.F90)

INTERFACE:

```
subroutine agrmet_georank_filename(name,dir,hemi,yr,mo,da,hr)  
    implicit none
```

ARGUMENTS:

```
integer, intent(in) :: hemi  
integer, intent(in) :: yr,mo,da,hr  
character*100      :: name  
character*50       :: dir
```

DESCRIPTION:

This routine generates the name of the GEO rank file by appending the hemisphere and timestamps to the root directory.

The arguments are:

hemi index of the hemisphere (1-NH, 2-SH)
dir full path to the directory containing the data
name created filename
yr 4 digit year
mo integer value of month (1-12)
da day of the month
hr hour of the day

29.1.82 calest (Source File: calest.F90)

REVISION HISTORY:

```
1 jun 89 initial version.....mr moore/sddc(agromet)  
11 feb 91 added use of precip probabilities in adjustment of  
precip coef prior to calculation of final precip  
estimate. eliminated use of cld amts in same process.  
resized pcoef and pcout arrays. isolated debug stmt....  
.....mr moore/sddc(agromet)  
4 may 91 chgd if stmts to allow ra, prcpwe, and rpcoeff precip  
rates of 0.0 to be used. allowed cldtyps of 0 to be  
processed, changed prob array to allow for cldtyp of  
0, updated the prolog.....capt bertone/sddc(agromet)  
3 mar 92 made chgs necessary to include cldamt as an index of  
pcoef array. eliminated array pcout, use pcoef instead..  
updtd prolog.....mr moore/sys(agromet)  
1 jun 92 expanded pcoef array to include rh class and stability  
class as indices. added cloud amount class' to condense  
10 cld amts into 5 keeping pcoef array smaller. added
```

use of pwgt.....mr moore sys(agromet)
 1 feb 94 resized the pcoef array, deleting the stability class
 rh class, cloud type and lat band indices and adding
 cld top ht class and tropopause ht class indices. chgd
 use of precip probabilities, making them a function of
 cld amt only. started using pwgte rather than pwgt.
 grouped comments and updtd prolog.....
mr moore, capt bertone, sysm(agromet)
 25 apr 94 added 'files accessed' section as 'id'ed during the
 94-01 qaa.....capt bertone, sysm(agromet)
 15 jun 96 reduced 'e' array size from (64,64,12) to (64,64,4).
 replaced chk of ca with chk of rpcof and eliminated
 calc of pprob which is now done in parent routine.
 updated prolog and brought up to stds.....
mr moore sysm(agromet)
 07 mar 97 removed precip coefficient, probability, and tropopause
 cloud height calculations. added calculations for climo
 estimate. altered code to run more efficiently.
 brought code up to current sysm and afgwc standards.
capt andrus, ssgt mccormick/sysm
 24 mar 97 added 'source' array which sets each type of input
 precip estimate with a specified integer source flag.
 array values are set according to determined precedence.
ssgt miller/sysm
 19 may 97 redimensioned source array to write each 3-hour sourced
 estimate into its own file.....ssgt miller/sysm
 04 dec 97 added ability to determine source of bogus, rtneph,
 and climo estimates. modified to allow only singular
 estimate type per grid point rather than a combined
 estimate. updated prolog and brought up to standards....
capt andrus/dnxm(agromet)
 08 oct 99 ported to ibm sp-2, updated prolog, incorporated
 FORTRAN 90 features, included the geo_precip estimate
 into the heirarchy.....capt hidalgo/agrmet
 01 jun 00 incorporated grnk values for geo_precip and used it to
 sort the order in which estimates will be used. Removed
 estimated weights from the sort.....ssgt campbell/agrmet
 21 feb 01 reduced third dimensions of all estimate arrays
 (number of three hourly time periods) from 4 to 2
 so module precip can run in 6 hourly cycles. changed
 variable name e to estpcp.....mr gayno/dnxm
 10 jun 02 modified references and variable names to reflect
 change from rtneph to cdafs2.....mr gayno/dnxm
 3 nov 05 Sujay Kumar, incorporated into LIS

INTERFACE:

```

subroutine calest( cest, cdafs2est, estpcp, gest, k, land, prcpwe,&
  quad9r, ra, source, hemi, imax, jmax, grnk )
  
```

```

  implicit none
  
```

ARGUMENTS:

integer, intent(in)	:: imax
integer, intent(in)	:: jmax
integer, intent(in)	:: grnk(imax,jmax)

```

integer, intent(in)          :: hemi
integer, intent(in)          :: k
integer, intent(in)          :: land(imax,jmax,2)
integer, intent(inout)        :: source(imax,jmax,2)
real, intent(in)             :: cest(imax,jmax)
real, intent(inout)           :: estpcp(imax,jmax,2)
real, intent(in)              :: gest(imax,jmax)
real, intent(in)              :: prcpwe(imax,jmax,2)
real, intent(in)              :: quad9r
real, intent(in)              :: ra(imax,jmax)
real, intent(in)              :: cdfs2est(imax,jmax,2)

```

DESCRIPTION:

calculate a 'final' 3-hour estimated precipitation amount based upon 3-hour precipitation estimate inputs, using a hierarchy also to identify the source of the estimate placed in the 'final' 3-hour estimated precipitation array.

Method

- determine the precedence given by the hierarchical values.
- for each gridpoint, check which estimates are available and place the highest ranked estimate in the estimation field.
- identify the source of the estimate.

The arguments and variables are:

cdfs2est precip est based on cdfs2 cloud amounts (mm/3hr)

cest precip est based purely on climatology (mm/3hr)

ehit used to determine valid points

estpcp final 3-hour estimated precip amt (mm/3hr)

gest precip est from geo-precip (mm/3hr)

grnk geo-precip ranking value (1-5)

1 = better than ssmi

2 = better than present weather

3 = better than cdfs2

4 = default value for geo-precip

5 = worse than climatological

hemi hemisphere (1 = nh, 2 = sh)

i loop counter, i-coordinate of the grid

imax number of gridpoints in the east/west direction

j loop counter, j-coordinate of the grid

jmax number of gridpoints in the north/south direction

k 3-hour time counter

l loop counter

land land/sea mask (0 = water, 1 = land)

order array to sort the desired hierarchical rank

prcpwe present weather estimated precip rate (mm/3hr)

quad9r parameter for 9999.0 value

ra ssmi rain rate (mm/3hr)

source array that contains estimate source information with specified source flags
 2 = ssmi est.
 3 = present weather est.
 4 = cdfs2 est.
 5 = geo_precip est.
 6 = climatological est.

temp dummy scalar

temp_order array used to hold shuffled values of order array

29.1.83 phsrel12 (Source File: phsrel12.F90)

REVISION HISTORY:

```

21 feb 01 initial version based on original routine phsrel.....
.....mr gayno/dnxm
10 jun 02 modified to reflect change from rtneph to cdfs2 data...
.....mr gayno/dnxm
3 nov 05 Sujay Kumar, incorporated into LIS

```

INTERFACE:

```

subroutine phsrel12( p12, j6hr, imax, jmax,&
    hemi, estpcp, source, cdfs2est,&
    land, quad9r, obswch, p3x, pathpcp)

```

USES:

```

use lis_logmod, only : logunit
use lis_fileIOMod, only : putget
use spmdMod
use listime_mgr, only : julhr_date

implicit none

```

ARGUMENTS:

```

integer,      intent(in)    :: hemi
integer,      intent(in)    :: imax
integer,      intent(in)    :: j6hr
integer,      intent(in)    :: jmax
integer,      intent(in)    :: land(imax,jmax,2)
integer,      intent(in)    :: oswch
real,         intent(inout) :: cdfs2est(imax,jmax,2)
real,         intent(inout) :: estpcp(imax,jmax,2)
real,         intent(out)   :: p3x(imax,jmax,2)
real,         intent(in)    :: p12(imax,jmax)
real,         intent(in)    :: quad9r

```

```
integer,         intent(inout) :: source(imax,jmax,2)
character*100, intent(in)      :: pathpcp
```

DESCRIPTION:

to parse 12-hourly real observed precipitation amounts into 3-hour amounts, and to write all precip arrays to file.

Method

- fill the 3 hrly observed output array with 9999.0.
- read in estimates from 6 and 9 hours ago.
- call parse routine to divide up the 12-hrly real precip (rain gauge) amounts into 3-hour precip amounts.
- write out the 3-hour parsed real precip data (over the 12 hour period) to file.
- write out the 3-hour estimated precip amounts to file.
- write out the 3-hour estimate sources to file.
- write out the 3-hour cdfs2-based precip estimates to file.

The arguments and variables are:

cdfs2est cdfs2-based precip estimate (mm/3hr)

e array of precip estimates (4 - 3hrly periods)

estpcp array of precip estimates (last 2 3hrly periods)

hemi hemisphere

imax number of gridpoints in east/west direction

jmax number of gridpoints in north/south direction

j6hr current 6-hrly start time (julian hour)

land point processing switches

obswch observation processing control switch (1 - use obs)

p3 3-hrly parsed real precip amounts (mm)

p12 12-hrly real (rain gauge) precip amounts (mm)

quad9r scalar value of 9999.0

source array that contains source-of-the-estimate information with specified source flags

pathpcp Path to agrmet precip files

ifil input file name including directory path

ofil output file name including directory path

The routines invoked are:

parse12 (29.1.84)

29.1.84 parse12 (Source File: parse12.F90)

REVISION HISTORY:

```
15 jun 96 initial version.....mr moore sysm(agromet)
 8 oct 99 ported to ibm sp-2, updated prolog, incorporated
   FORTRAN 90 features.....capt hidalgo/agrmet
21 feb 01 renamed routine parse12 (formally parse). removed
   calculation of "ratio" as this field was not being
   used.....mr gayno/dnxm
 3 nov 05 Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine parse12( hemi, land, p12, e, p, imax, jmax )

implicit none
```

ARGUMENTS:

integer, intent(in)	:: hemi
integer, intent(in)	:: imax
integer, intent(in)	:: jmax
integer, intent(in)	:: land(imax,jmax,2)
real, intent(in)	:: e(imax,jmax,4)
real, intent(out)	:: p(imax,jmax,4)
real, intent(in)	:: p12(imax,jmax)

DESCRIPTION:

to break-up 'observed' 12-hrly real amounts into 3-hrly real amounts.

loop thru the grid points. for each major landmass point with a valid 12-hrly observed real precip amount...
a. build a 12-hrly estimated precip amount from the 3-hrly estimated amounts in the four 3-hrly periods.
b. if the 12-hrly estimated precip amount = 0.0 break the 12-hrly real amount into four equal parts.
c. if the 12-hrly estimated precip amount $>$ 0.0 break the 12-hrly real amount into four amounts, each determined by the ratio of the 3-hrly estimated amount to the 12-hrly estimated amount.

The arguments and variables are:

e 3-hrly estimated precip amounts (mm)
e12 sum of 3-hrly precip estimates for the 12 hrs
hemi hemisphere
i loop index
imax number of gridpoints in east/west direction
j loop index
jmax number of gridpoints in north/south direction
k loop index
land point processing switches
p 3-hrly parsed real precip amounts (mm)
p12 12-hrly real precip amounts (mm)

29.1.85 phsrel6 (Source File: phsrel6.F90)

REVISION HISTORY:

```
21 feb 01 initial version based on original routine phsrel.....  
.....mr gayno/dnxm  
10 jun 02 modified to reflect change from rtneph to cdfs2 data...  
.....mr gayno/dnxm  
3 nov 05 Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine phsrel6 ( estpcp, hemi, j6hr, land, p6,&  
quad9r, cdfs2est, source, &  
imax, jmax, obswch, p3, pathpcp)
```

USES:

```
use lis_logmod, only : logunit  
use lis_fileIOMod, only : putget  
use spmdMod  
use listime_mgr, only : julhr_date  
  
implicit none
```

ARGUMENTS:

integer, intent(in)	:: hemi
integer, intent(in)	:: imax
integer, intent(in)	:: jmax
integer	:: j3hr
integer, intent(in)	:: j6hr
integer	:: k
integer, intent(in)	:: land(imax,jmax,2)
integer, intent(in)	:: obswh
integer, intent(inout)	:: source(imax,jmax,2)
real, intent(inout)	:: cdfs2est(imax,jmax,2)
real, intent(inout)	:: estpcp(imax,jmax,2)
real, intent(out)	:: p3(imax,jmax,2)
real, intent(in)	:: p6(imax,jmax)
real, intent(in)	:: quad9r
character*100, intent(in)	:: pathpcp

DESCRIPTION:

to parse 6-hourly real observed precipitation amounts into 3-hour amounts, and to write all output arrays to file.

Method

- fill the 3 hrly observed output array with 9999.0.
- call parse routine to divide up the 6-hrly real precip (rain gauge) amounts into 3-hour precip amounts.
- write out the 3-hour parsed real precip amounts to file.
- write out the 3-hour estimated precip amounts to file.
- write out the 3-hour estimate sources to file.
- write out the 3-hour cdfs2-based precip estimates to file.

The arguments and variables are:

cdfs2est cdfs2-based precip estimate (mm/3hr)
estpcp 3-hrly estimated precip amounts (mm)
hemi hemisphere
imax number of gridpoints in east/west direction
jmax number of gridpoints in north/south direction
j6hr current 6-hrly start time (julian hour)
j3hr current 3-hrly time (julian hour)
k loop index, 'i'th 3-hrly time period
land point processing switches
obswch observation processing control switch (1 - use obs)
p3 3-hrly parsed real precip amounts (mm)
p6 6-hrly real (rain gauge) precip amounts (mm)
quad9r scalar value of 9999.0
source array that contains source-of-the-estimate information with specified source flags
pathpcp Path to agrmet precip files
The routines invoked are:
parse6 (29.1.86)

29.1.86 parse6 (Source File: parse6.F90)

REVISION HISTORY:

```

21 feb 01 initial version based on original parse routine.....
.....mr gayno/dnxm
4 nov 05 incorporated into LIS, sujay kumar

```

INTERFACE:

```

subroutine parse6( hemi, land, p6, estpcp, p, imax, jmax )

implicit none

integer, intent(in)          :: hemi
integer                      :: i
integer, intent(in) :: imax
integer                      :: j
integer, intent(in) :: jmax
integer :: k
integer, intent(in)          :: land(imax,jmax,2)

real,    intent(in)          :: estpcp(imax,jmax,2)

```

```

real :: e6
real, intent(out) :: p(imax,jmax,2)
real, intent(in)  :: p6(imax,jmax)

```

DESCRIPTION:

to break-up 'observed' 6-hrly real rain gauge amounts into 3-hrly amounts.

loop thru the points for a hemisphere. for each land point with a valid 6-hrly observed real precip amount...
 a. build a 6-hrly estimated precip amount from the 3-hrly estimated amounts in the two 3-hrly periods.
 b. if the 6-hrly estimated precip amount = 0.0 break the 6-hrly real amount into two equal parts.
 c. if the 6-hrly estimated precip amount > 0.0 break the 6-hrly real amount into two amounts, each determined by the ratio of the 3-hrly estimated amount to the 6-hrly estimated amount.

The arguments and variables are:

estpcp 3-hrly estimated precip amounts (mm)

e6 sum of 3-hrly precip estimates for the 6-hrly period

hemi hemisphere

i loop index

imax number of gridpoints in east/west direction

j loop index

jmax number of gridpoints in north/south direction

k loop index

land point processing switches

p 3-hrly parsed real (rain gauge) precip amounts (mm)

p6 6-hrly real (rain gauge) precip amounts (mm)

29.1.87 agrmet_valid (Source File: agrmet_valid.F90)

REVISION HISTORY:

15 jun 96	initial version.....mr moore/sysm(agromet)
10 apr 97	brought up to software standards. modified to create supergrid of new source files. also now begins summing 12-hr estimate and real quad9r values while in box arrays for greater cpu and memory efficiency.....ssgt miller/sysm
02 may 97	changed array indices to prevent 'thrashing' of memory. added check of source value to ensure validity of est. updated prolog and brought up to standards.....capt andrus/sysm
7 oct 99	ported to ibm sp-2, updated prolog, incorporated FORTRAN 90 features.....capt hidalgo/agrmet
05 jan 01	renamed variables to reflect 6-hourly change to 6-hourly cycles.....mr gayno/dnxm

```
10 jun 02 modified to reflect change from rtneph to cdfsii.....
.....mr gayno/dnxm
3 nov 05 Sujay Kumar, Initial Code
```

INTERFACE:

```
subroutine agrmet_valid( hemi, pcap, mrg, est, est6, src,&
    rel, rel6, cdfsii3, cdfsii6, srcwts, julhr,&
    imax, jmax ,pathpcp)
```

USES:

```
use lis_logmod, only : logunit
use lis_fileIOMod, only : putget
use listime_mgr, only : julhr_date

implicit none
```

ARGUMENTS:

```
integer,      intent(in)      :: hemi
integer,      intent(in)      :: imax
integer,      intent(in)      :: jmax
integer          :: src(imax,jmax)
real,         intent(inout)   :: cdfsii3(imax,jmax)
real,         intent(out)     :: cdfsii6(imax,jmax)
real          :: est(imax,jmax)
real,         intent(out)     :: est6(imax,jmax)
real,         intent(out)     :: mrg(imax,jmax)
real          :: rel(imax,jmax)
real,         intent(out)     :: rel6(imax,jmax)
real,         intent(in)      :: pcap
real,         intent(in)      :: srcwts(8)
integer,      intent(in)      :: julhr
character*100, intent(in)     :: pathpcp
```

DESCRIPTION:

to validate the 3 hrly estimated and phased real precip amounts and set (or sum) into the 6 hrly values. to set the mrgd value.

Method

1. loop thru points in the hemisphere
- a. if 3 hrly estimated and/or real value passes validity checks
 - 1) put (or sum) the value into 6 hrly value
 - 2) put 3 hrly real value into 3 hrly mrg value
- b. else
 - 1) put 9999.0 into 3 hrly estimated and/or real value
 - 2) put 3 hrly estimated value into mrg value

The arguments and variables are:

cdfsii3 3hr cdfsii based precip estimate (mm/3hr)

cdfsii6 6hr cdfsii based precip estimate (mm/6hr)

hemi current hemisphere (1 = nh, 2= sh)

est estimated 3hr precip array (mm/3hr)
est6 estimated 6hr precip array (mm/6hr)
i loop index, point's i-coordinate
ifil character array holding path/name of input file(s)
imax number of gridpoints in east-west direction
j loop index, point's j-coordinate
jmax number of gridpoints in north-south direction
rel parsed 3hr real precip array (mm/3hr)
rel6 parsed 6hr real precip array (mm/6hr)
hemi hemisphere (1=north, 2=south)
mrg merged 3hr precip array (mm/s = kg/(m² s))
pathpcp character array for path to precip files
pcap precip cap limit (mm/3hr)
quad8r integer constant (9998.0) indicating bad data
quad9r integer constant (9999.0) indicating bad data
src precip data source array
srcwts value weight placed on each source

29.1.88 pcp_barnes (Source File: pcp_barnes.F90)

REVISION HISTORY:

```

15 jun 96 initial version.....mr moore sysm(agromet)
10 apr 97 brought up to software standards. combined barn1,
          barn2, and blend into one routine. made use of source
          type by using different spread weights and radius based
          upon source type.....ssgt miller/sysm
02 may 97 changed order of array indices to prevent 'thrashing'
          in memory. updated prolog and brought up to standards.
          .....capt andrus/sysm(agromet)
7 oct 99 ported to ibm sp-2, updated prolog, incorporated
          FORTRAN 90 features.....capt hidalgo/agrmet
31 aug 01 corrected error which assigned wrong weights and
          radii to the wrong precipitation source. geoprecip
          and climo were reversed as were pres/past wx and
          ssm/i. added code to use addrad(7) and (8) which were
          not being used. made corresponding fixes to
          control.spread file.....mr gayno/dnxm
3 nov 05 Sujay Kumar, Initial Code

```

INTERFACE:

```
subroutine pcp_barnes( mrgp, radius, addrad, maxrad, minrad, src, &
srcwts, varrad, land, tmp, wgt, imax, jmax )
```

USES:

```
use lis_logmod, only : logunit  
implicit none
```

ARGUMENTS:

integer,	intent(in)	:: addrad(8)
integer,	intent(in)	:: imax
integer,	intent(in)	:: jmax
integer,	intent(in)	:: land(imax,jmax)
integer,	intent(in)	:: maxrad(6)
integer,	intent(in)	:: minrad(6)
integer,	intent(in)	:: radius(imax,jmax)
integer,	intent(in)	:: src(imax,jmax)
integer,	intent(in)	:: varrad(4)
real,	intent(inout)	:: mrgp(imax,jmax)
real,	intent(in)	:: srcwts(8)
real,	intent(out)	:: tmp(imax,jmax)
real,	intent(inout)	:: wgt(imax,jmax)

DESCRIPTION:

to perform a barnes analysis on the precipitation data.

Method

1. loop thru pts in the hemisphere
 - a. if pt's parsed real precip amt is valid ...
 - 1) set a local scan area
 - 2) calc a local mean parsed real precip amt
 - 3) eval amt diff btwn local mean and current pt adjust initial spread radius accordingl b. set final spreading radius for each precip source
 2. initialize variables to zero
 3. loop thru pts in the hemisphere
 - a. set the spreading radius in grid dist's.
 - b. determine the surrounding pts to be affected by the amt at the pt (the spread pt).
 - c. for each valid surrounding pt...
 - 1) calc the pt's dist from the spread pt.
 - 2) if the pt is within the spread radius, calc and accum the spread wgt and a weighted 3-hrly merged precip amts.
 4. again, loop thru the pts in the hemisphere...
if a precip amt was spread to this pt, calc a new 3-hrly merged precip value for the pt using the accumulated spread wgt and the accum weighted 3-hrly merged precip amt.

The arguments and variables are:

a dummy variable

addrad added radius value for barnes analysis

b dummy variable

cnt cnumber of points to which values were spread

idis integer distance from pt to surrounding pts
iend end point for ii loop
ii loop index for local spread area
imax number of gridpoints in east-west direction
irad radius of the observation points
is loop index for supergrid
istrt start point for ii loop
jend end point for jj loop
jj loop index for local spread area
jmax number of gridpoints in north-south direction
js loop index for supergrid
jstrt start point for jj loop
land supergrid point processing switches
maxi upper i coordinate value
maxj upper j coordinate value
maxrad maximum radius value
meanv mean value of other (local) precip amts
mini lower i coordinate value
minj lower j coordinate value
minrad minimum radius value
mrgp merged precip amount
numval counter
quad8r integer constant (9998.0) indicating bad data
quad9r integer constant (9999.0) indicating bad data
rad distance from (i,j) to (is,js)
radius supergrid integer radius for obs spreading
ri real version of i
rj real version of j
rrad supergrid real scan radius for obs spreading
rsqd distance squared from (i,j) to (is,js)
sigmav std deviation of surrounding obs values
src source amount gridpoints
srcdis source distance of the points

srcwt weighted source
srcwts value weight placed on each source
sumsqqr sum of squares statement function
sumval accumulation variable
tmp temporary array to hold weighted values
varrad radius change amount due to the real precip variance
vdiff abs value of diff btwn srel and meanv
wgt weighted array point values
wt 1.0 to .018 weight applied to i/j
x dummy variable
y dummy variable
ztest variable that holds proper spread weight

29.1.89 make03 (Source File: make03.F90)

REVISION HISTORY:

```

15 jun 96 initial version.....mr moore/sysm
10 apr 97 brought up to software standards. modified to more
            efficiently convert supergrid merged precip array into
            box arrays, write to file, and sum into 12-hr merged
            amount.....ssgt miller/sysm
02 may 97 corrected error in output caused by dividing 12-hour
            merged precipitation values by 10800. updated prolog
            and brought up to standards.....capt andrus/sysm
7 oct 99 ported to ibm sp-2, updated prolog, incorporated
            FORTRAN 90 features. modified the part of code that
            divides the merged values by 10800 so that only the
            values needed by FLUX3 (the 3hrly merged values) are
            divided by 10800 (to get the units in terms of a flux)
            .....capt hidalgo/agrmet
05 jan 01 modified variable names to reflect 6-hourly cycles....
            .....mr gayno/dnxm
3 nov 05 Sujay Kumar, Initial Code

```

INTERFACE:

```
subroutine make03( mrgp, mrgp6,imax, jmax )
```

```
implicit none
```

ARGUMENTS:

```

integer,      intent(in)    :: imax
integer,      intent(in)    :: jmax
real,        intent(inout)  :: mrgp(imax,jmax)
real,        intent(inout)  :: mrgp6(imax,jmax)

```

DESCRIPTION:

to write the 3-hrly merged precip amounts to file and accumulate the 3-hrly merged precip amounts into a 6-hrly merged precip total.

Method

1. loop thru grid points in the hemisphere - sum the 3-hrly amts into 6-hrly amts

The arguments and variables are:

i loop index

imax number of gridpoints in east/west direction

j loop counter

jmax number of gridpoints in north/south direction

mrgp supergrid 3-hrly merged precip amounts (mm)

mrgp6 supergrid 6-hrly merged precip amounts (mm)

29.1.90 julhr_date10 (Source File: julhr_date10.F90)

REVISION HISTORY:

```
15 oct 1998 initial version.....mr moore/dnxm
10 aug 1999 ported to ibm sp2. added intent attributes to
arguments.....mr gayno/dnxm
29 oct 2005 Sujay Kumar, Adopted in LIS
```

INTERFACE:

```
subroutine julhr_date10( julhr, date10)
```

USES:

```
use listime_mgr, only : tmjul4
use lis_logmod, only : lis_abort

implicit none
```

ARGUMENTS:

```
character*10,    intent(out)  :: date10
integer,         intent(in)   :: julhr
```

DESCRIPTION:

to convert from a julian hour to a 10 digit date/time group (yyyymmddhh)

Method

- call utility routine tmjul4 to convert from julian hours to year, month, day, hour.
- perform several checks to ensure date information passed back from tmjul4 is valid.
- if date is good, convert from integer data to a 10 digit date/time group and pass back to calling routine.

The arguments are variables are:

date10 output 10-digit character date time group (yyyymmddhh)

dd day of the month

hh time of day in hours

j loop counter

julhr input julian hour

mm month of the year

yyyy four digit year

The routines invoked are:

tmjul4 (5.4.25)

convert julian hour to hour,day,month and year

29.1.91 time_interp_agrmet (Source File: time_interp_agrmet.F90)

REVISION HISTORY:

25 Jul 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine time_interp_agrmet(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use baseforcing_module, only : lisforc
use listime_mgr
use spmdMod
use agrmetdomain_module, only : agrmet_struct
```

29.1.92 agrmetforcing_finalize (Source File: agrmetforcing_finalize.F90)

REVISION HISTORY:

25 Oct 2005; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine agrmetforcing_finalize
```

USES:

```
use agrmetdomain_module
use lisdrv_module, only : lis
```

DESCRIPTION:

Routine to cleanup allocated structures for AGRMET forcing.

30 BERG

This section describes the implementation of the bias-corrected atmospheric reanalysis data developed by Aaron Berg. The data is global 0.5 degree in latlon projection.

30.1 Fortran: Module Interface `bergdomain_module` (Source File: `bergdomain_module.F90`)

This module contains variables and data structures that are used for the implementation of the bias-corrected atmospheric reanalysis data (Berg et al. 2003). The data is global 0.5 degree dataset in latlon projection, and at 6 hourly intervals. The derived data type `berg_struct` includes the variables that specify the runtime options, and the weights and neighbor information to be used for spatial interpolation. They are described below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

fmodeltime1 The nearest, previous 6 hour instance of the incoming data (as a real time).

fmodeltime2 The nearest, next 6 hour instance of the incoming data (as a real time).

remask1d The data used to remask the input data to the LIS mask.

bergdir Directory containing the input data

emaskfile File containing the 0.5 deg land-sea mask used in the input data.

elevfile File with the elevation definition for the input data.

mi Number of points in the input grid

rlat1 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

rlon1 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

n11,n121,n211,n221 Arrays containing the neighbor information of the input grid for each grid point in LIS, for bilinear interpolation.

w111,w121,w211,w221 Arrays containing the weights of the input grid for each grid point in LIS, for bilinear interpolation.

rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

n12,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

Berg, A.A., J.S. Famiglietti, J.P. Walker, and P.R. Houser, 2003: Impact of bias correction to reanalysis products on simulations of North American soil moisture and hydrological fluxes. *Journal of Geophysical Research*, 108, 4490, DOI: 10.1029/2002JD003334.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
-----  
public :: defineNativeBERG      !defines the native resolution of  
                                !the input data  
-----
```

PUBLIC TYPES:

```
-----  
public :: berg_struct  
-----
```

30.1.1 defineNativeBERG (Source File: bergdomain_module.F90)

REVISION HISTORY:

26Jan2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine defineNativeBERG()
```

USES:

```
use lisdrv_module, only : lis  
implicit none
```

DESCRIPTION:

Defines the native resolution of the input forcing for BERG data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readbergcrd (30.1.2)
reads the runtime options specified for BERG data

readbergmask (30.1.4)
reads the 0.5 degree land sea mask

bilinear_interp_input (7.0.2)
computes the neighbor, weights for bilinear interpolation

conserv_interp_input (7.0.4)
computes the neighbor, weights for conservative interpolation

read_berg_elev (30.1.3)
reads the native elevation of the berg data to be used for topographic adjustments to the forcing

30.1.2 readbergcrd (Source File: readbergcrd.F90)

REVISION HISTORY:

26Jan2004; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine readbergcrd()
```

USES:

```
use lis_logmod, only : logunit
use lisdrv_module, only : lis, config_lis
use LIS_ConfigMod
use bergdomain_module, only : berg_struc
```

DESCRIPTION:

This routine reads the options specific to BERG forcing from the LIS configuration file.
The routines invoked are:

LIS_ConfigGetAttribute (5.7.5)
read the specified attribute

LIS_ConfigFindLabel (5.7.8)
find the specified label to read the attribute

30.1.3 read_berg_elev (Source File: read_berg_elev.F90)

REVISION HISTORY:

17Dec2004; Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_berg_elev(n)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use baseforcing_module, only : lisforc
use bergdomain_module, only : berg_struc
use lis_logmod, only : logunit
```

30.1.4 readbergmask (Source File: readbergmask.F90)

REVISION HISTORY:

16 Apr 2002: Urszula Jambor; Initial Code
24 Nov 2003: Sujay Kumar; Included in LIS

INTERFACE:

```
subroutine readbergmask(n)
```

USES:

```
use lisdrv_module, only :lis      ! LIS non-model-specific 1-D variables
use bergdomain_module, only : berg_struc
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Reads in land-sea mask for 0.5 degree Berg (Reanalysis ECMWF) data set, changes grid from ECMWF convention to LIS convention by calling `berggrid_2.lisgrid`, and transforms grid array into 1D vector for later use. . The arguments are:

n index of the nest

The routines invoked are:

berggrid_2.lisgrid (30.1.7)
transform the BERG data to the LIS grid

30.1.5 getberg (Source File: getberg.F90)

REVISION HISTORY:

```
11 Apr 2002: Urszula Jambor; original code based on getgeos.f
22 Oct 2002: Urszula Jambor; Limited SW forcing processing to
              land-only grid points
24 Nov 2003: Sujay Kumar; Included the scheme in LIS
```

INTERFACE:

```
subroutine getberg(n)
```

USES:

```
use lisdrv_module, only : lis
use baseforcing_module, only : lisforc
use bergdomain_module, only : berg_struc
use listime_mgr
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates 6-hrly, 1/2 degree Reanalysis ECMWF forcing. At the beginning of a simulation, the code reads the most recent past data (nearest 6 hour interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. The strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day . The arguments are:

n index of the nest

The routines invoked are:

tick (5.4.22)
call to advance or retract time

retberg (30.1.6)
call to read the berg data and perform spatial interpolation

30.1.6 retberg (Source File: retberg.F90)

REVISION HISTORY:

09 Apr 2002: Urszula Jambor; Original code, based on readgeos.f
20 Dec 2002: Matt Rodell; Don't ipolate if running 1/2 deg, also fix
error in setting v-wind to zero
25 Mar 2003: Urszula Jambor; Modified argument list passed to GEOGILL2.
24 Nov 2003; Sujay Kumar; Included in LIS
15 Mar 2004; Matt Rodell; Fix test for beginning of restart run

INTERFACE:

```
subroutine retberg(order, n, yr, mon, da, hr, ferror)
USES:
use lisdrv_module, only : lis,lisdom
use baseforcing_module, only: lisforc
use bergdomain_module, only : berg_struc
use spmdMod, only : iam,masterproc
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: order
integer, intent(in)    :: n
integer, intent(in)    :: yr
integer, intent(in)    :: mon
integer, intent(in)    :: da
integer, intent(in)    :: hr
integer, intent(inout) :: ferror
```

DESCRIPTION:

For the given time, reads 8 parameters from 1/2 degree Reanalysis ECMWF data, transforms into 9 LIS forcing parameters and interpolates to the LIS domain.

Reanal. ECMWF FORCING VARIABLES available every 6 hours:

1. 2T 2 metre air temperature [K], instantaneous

2. 2D 2 metre dew point temperature [K], instantaneous
3. SSRD Downward shortwave flux, surface [W/m²s], accumulation
4. STRD Downward longwave radiation, surface [W/m²s], accumulation
5. WIND Calculated absolute 10 metre wind speed [m/s], instantaneous
6. P Calculate surface pressure [Pa], instantaneous
7. LSP+CP Total precipitation [m], accumulation
8. CP Convective precipitation [m], accumulation

order flag indicating which data to be read (order=1, read the previous 6 hourly instance, order=2, read the next 6 hourly instance)

n index of the nest

yr current year

mon current month

da current day of the year

hr current hour of day

ferror flag to indicate success of the call (=0 indicates success)

The routines invoked are:

berggrid_2_lisgrid (30.1.7)

transform the BERG data to the LIS grid

bilinear_interp (7.0.3)

spatially interpolate the forcing data using bilinear interpolation

conserv_interp (7.0.5)

spatially interpolate the forcing data using conservative interpolation

geogfill2 (30.1.10)

fill in any grid points due to mismatched grids.

30.1.7 berggrid_2_lisgrid (Source File: **retberg.F90**)

REVISION HISTORY:

10 Apr 2002: Urszula Jambor; Code adapted from
ecmwfgrid_2_grid2catgrid, by R. Reichle

INTERFACE:

```
subroutine berggrid_2_lisgrid( nx, ny, grid_data )
  implicit none
```

ARGUMENTS:

```
integer, intent(in)          :: nx, ny
real, intent(inout), dimension(nx,ny) :: grid_data
```

DESCRIPTION:

Changes grid data from ECMWF data convention to LIS convention
ECMWF: North-to-South around Greenwich Meridian Global grid. Data are written as flat binary from "upper left to lower right" starting at 0.5-degree grid point center coordinates: 0.25E,89.75N and going to 0.25W,89.75S. Here is the write statement:

```
do i = 1,360
    write(14) (val(j,i),j=1,720)
end do
```

LIS: South-to-North around Date Line Full global grid. Starts at the southernmost latitude and date line, going east and then north.

30.1.8 fillgaps (Source File: retberg.F90)

Fills in values for NSIPP tilespace land points where no ECMWF reanalysis data is available via GEOG-FILL by assigning most appropriate land-point value along the latitudinal circle of original tilespace point. Developed manually with 17 points in mind.

REVISION HISTORY:

```
23 Jul 2002: Urszula Jambor
03 Sep 2002: Urszula Jambor, revised points to reflect
              NSIPP land mask correction.
```

INTERFACE:

```
subroutine fillgaps( nc, nr, v, arr )
```

30.1.9 time_interp_berg (Source File: time_interp_berg.F90)

REVISION HISTORY:

```
24 Nov 2003: Sujay Kumar; Initial version of code adapted from the GLDAS
              version
```

INTERFACE:

```
subroutine time_interp_berg(n)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use baseforcing_module, only : lisforc
use listime_mgr
use bergdomain_module, only : berg_struc
use spmdMod
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep. Downward shortwave radiation is interpolated using a zenith-angled based approach. Precipitation and longwave radiation are not temporally interpolated, and the previous 6 hourly value is used. All other variables are linearly interpolated between the 6 hourly blocks.

The routines invoked are:

time2date (5.4.21)

converts the time to a date format

zterp (5.26.5)

zenith-angle based interpolation

30.1.10 geogfill2 (Source File: geogfill2.F90)

REVISION HISTORY:

20 Sep 2002: Urszula Jambor; Modified original geogfill routine
for use with Reanalysis forcing sets prepared by
Aaron Berg via NSIPP

29 Jan 2003: Urszula Jambor; Removed LDAS & GRID modules from list of
arguments, instead pass only needed variables directly
(nc, nr, fmask).

27 Jan 2004: Matt Rodell; Make sure all points are filled, rename
data array to fdata.

INTERFACE:

```
subroutine geogfill2(n, nc, nr, geogmask, fdata, v, tmask)
```

USES:

```
use lisdrv_module, only : lisdom  
  
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n  
integer :: nc  
integer :: nr  
logical*1 :: geogmask(nc, nr)  
real :: fdata(nc, nr)  
integer :: v  
logical*1 :: tmask(nc, nr)
```

DESCRIPTION:

Fill in grid points that are invalid in LIS due to differences in geography between forcing and land surface. Based on original geogfill.F90 for use with Reanalysis forcing fdata, to account for differences in land masks. NOTE** for Reanalysis fdata, v=6 is the V-component of wind, which is always set to ZERO since the U-component (v=5) is assigned the absolute value of wind speed.

For v=1:temperature, v=2:specific humidity, v=4:LW radiation, v=5:wind, and v=7:pressure, data values of zero are not allowed to contribute to average fill-value. For v=3:SW radiation, v=8,9:precipitation, use

the mask defined by valid temperature data points to establish whether to seek fill-value for given column and row, rather than include vs exclude zeroes.

The arguments are:

n index of the nest
nc number of columns of the grid (along the east-west dimension)
nr number of rows of the grid (along the north-south dimension)
geogmask interpolated forcing bitmap
fdata interpolated forcing
v index of the variable
tmask valid temperature points (used as the land mask)

30.1.11 `bergforcing_finalize` (Source File: `bergforcing_finalize.F90`)

REVISION HISTORY:

25 Oct 2005; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine bergforcing_finalize
```

USES:

```
use lisdrv_module, only : lis
use bergdomain_module
```

DESCRIPTION:

Routine to cleanup allocated structures for berg forcing.

31 ECMWF

The operational, global analysis products from the European Center for Medium-Range Weather Forecasts (ECMWF) is available on a T_L511 triangular truncation, linear reduced gaussian grid for four synoptic hours: 00, 06, 12, and 18 UTC, which are used in LIS. The radiation products from the data assimilation system at the ECMWF is based on the scheme simulated after Morcrette's work [13]. The estimation of shortwave and longwave radiation includes schemes to include effects of adsorption by water vapor and other gases and aerosol scattering based on different parameterizations [18, 12].

31.1 Fortran: Module Interface `ecmwfdomain_module` (Source File: `ecmwfdomain_module.F90`)

This module contains variables and data structures that are used for the implementation of the ECMWF forcing data. The data is global 0.25 degree dataset in latlon projection, and at 3 hourly intervals. The derived data type `ecmwf_struct` includes the variables that specify the runtime options, and the weights and neighbor information to be used for spatial interpolation. They are described below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data
nmif Number of forcing variables in the ECMWF data
ecmwftime1 The nearest, previous 3 hour instance of the incoming data (as a real time).
ecmwftime2 The nearest, next 3 hour instance of the incoming data (as a real time).
ecmwfdir Directory containing the input data
mi Number of points in the input grid
rlat1 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)
rlon1 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)
n11,n121,n211,n221 Arrays containing the neighbor information of the input grid for each grid point in LIS, for bilinear interpolation.
w111,w121,w211,w221 Arrays containing the weights of the input grid for each grid point in LIS, for bilinear interpolation.
rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)
rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)
n12,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.
w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: defineNativeECMWF      !defines the native resolution of
!the input data
```

PUBLIC TYPES:

```
public :: ecmwf_struct
```

31.1.1 defineNativeECMWF (Source File: ecmwfdomain_module.F90)

REVISION HISTORY:

11Dec2003: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine defineNativeECMWF
```

USES:

```
use lisdrv_module, only: lis
implicit none
```

DESCRIPTION:

Defines the native resolution of the input forcing for ECMWF data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readecmwfcrd (30.1.2)

reads the runtime options specified for ECMWF data

bilinear_interp_input (7.0.2)

computes the neighbor, weights for bilinear interpolation

conserv_interp_input (7.0.4)

computes the neighbor, weights for conservative interpolation

31.1.2 readecmwfcrd (Source File: readecmwfcrd.F90)

REVISION HISTORY:

11 Dec 2003; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine readecmwfcrd()
```

USES:

```
use lisdrv_module, only : lis, config_lis
use LIS_ConfigMod
use ecmwfdomain_module, only : ecmwf_struct
use lis_logmod, only : logunit
```

DESCRIPTION:

This routine reads the options specific to ECMWF forcing from the LIS configuration file.
The routines invoked are:

LIS_ConfigGetAttribute (5.7.5)

read the specified attribute

LIS_ConfigFindLabel (5.7.8)

find the specified label to read the attribute

31.1.3 getecmwf (Source File: getecmwf.F90)

REVISION HISTORY:

18 Jun 2003: Urszula Jambor; original code based on getreanlecmwf.F90
20 Feb 2006: Sujay Kumar; Modified with nesting options

INTERFACE:

```
subroutine getecmwf(n)
```

USES:

```
use lisdrv_module, only : lis
use baseforcing_module, only : lisforc
use listime_mngr
use ecmwfdomain_module, only : ecmwf_struct
use lis_logmod, only : logunit

implicit none

integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates 3-hrly, 1/4 degree ECMWF forcing. At the beginning of a simulation, the code reads the most recent past data (nearest 3 hour interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. The strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day . The arguments are:

n index of the nest

The routines invoked are:

tick (5.4.22)
call to advance or retract time

retecmaf (31.1.4)
call to read the ECMWF data and perform spatial interpolation

31.1.4 retecmaf (Source File: retecmaf.F90)

REVISION HISTORY:

18 Jun 2003: Urszula Jambor; original code

INTERFACE:

```
subroutine retecmaf( order, n, yr, mon, da, hr, ferror )
```

USES:

```
use lisdrv_module, only : lis,lisdom
use baseforcing_module, only : lisforc
use listime_mngr
use ecmwfdomain_module, only : ecmwf_struct
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)    :: n   ! lower(1) or upper(2) time interval bdry
integer, intent(in)    :: order ! lower(1) or upper(2) time interval bdry
integer, intent(in)    :: yr,mon,da,hr      ! data and hour (multiple of 3)
integer, intent(inout) :: ferror          ! set to zero if there's an error
```

DESCRIPTION:

For the given time, reads the parameters from 1/4 degree ECMWF data, transforms into 9 LIS forcing parameters and interpolates to the LIS domain.

ECMWF model output variables used to force LIS fall into 2 categories: $-_i$ inst3, Instantaneous values, available every 3 hours

$-_i$ accum, Time integrated values (accumulations), updated every 3 hours

NOTE-1: be aware that ECMWF outputs large-scale and convective precipitation separately. For total precipitation, need to sum the two fields, LSP+CP=TP. Reversed traditional order of model-forcing precip fields for easier processing – interchange at end of routine to preserve traditional order.

NOTE 2: only time2 SW flux accumulations used in interpolation

NOTE-3: read in of Albedo is currently suppressed. This field is instantaneous and available every 6 hours. At this time, all LIS LSMs replace this parameter.

ECMWF FORCING VARIABLES:

1. T inst3, near-surface temperature, 10 metres [K]
2. Q inst3, near-surface specific humidity, 10 metres[kg/kg]
3. SSRD accum, surface solar radiation downwards [$\text{W m}^{**-2} \text{s}$]
4. STRD accum, surface thermal radiation downwards [$\text{W m}^{**-2} \text{s}$]
5. U inst3, zonal wind, 10 metres [m/s]
6. V inst3, meridional wind, 10 metres[m/s]
7. SP inst3, surface pressure [Pa]
8. CP accum, convective precipitation [m]
9. LSP accum, large scale precipitation [m]

order flag indicating which data to be read (order=1, read the previous 3 hourly instance, order=2, read the next 3 hourly instance)

n index of the nest

yr current year

mon current month

da current day of the year

hr current hour of day

ferror flag to indicate success of the call (=0 indicates success)

The routines invoked are:

ret_inst3 (31.1.5)

retrieves the instantaneous variable

ret_accum (31.1.6)

retrieves the accumulated variable

interp_iv (31.1.7)

spatial interpolation from the ECMWF grid to the LIS grid

31.1.5 ret_inst3 (Source File: retecmwf.F90)

INTERFACE:

```
subroutine ret_inst3(dir,yr,mo,da,hr,necmwf,f,id,gridDesc,lb,fret)
implicit none
```

ARGUMENTS:

```
character(200), intent(in) :: dir
integer, intent(in)      :: yr,mo,da,hr,necmwf,id
integer, intent(out)     :: gridDesc(200)
real, intent(out)        :: f(necmwf)
logical*1, intent(out)   :: lb(necmwf)
integer, intent(inout)   :: fret
```

DESCRIPTION:

This routine opens the corresponding ECMWF data file to extract the specified variable, which represents an instantaneous value. Should be used for near-surface temperature, specific humidity, winds, and surface pressure.

The arguments are:

dir directory path name for ECMWF data

yr year

mo month

da day of month

hr hour of day

necmwf number of elements of input grid

f 1D input forcing field

id index indicating the forcing variable

gridDesc Array describing the input grid

lb unpacked input bitmap

fret return code (0-indicates success)

31.1.6 ret_accum (Source File: retecmwf.F90)

INTERFACE:

```
subroutine ret_accum(dir,yr,mo,da,hr,necmwf,f,id,gridDesc,lb,fret)
```

USES:

```
use listime_mgr, only : tick
use lis_logmod, only : logunit
implicit none
```

ARGUMENTS:

```
character(200) :: dir
integer :: yr,mo,da,hr
integer :: necmwf
real :: f(necmwf)
integer :: id
integer :: gridDesc(200)
logical*1 :: lb(necmwf)
integer :: fret
```

DESCRIPTION:

This routine opens the corresponding ECMWF data file to extract the specified variable, which represents an accumulated value. Should be used for radiation and precipitation. Returns field valid for 3-hour interval prior to given time, except for downward sw flux, where interval length varies. Returns forcing in units appropriate to LIS:
W/(m²s) to W/m²
m to mm/s

The arguments are:

dir directory path name for ECMWF data

yr year

mo month

da day of month

hr hour of day

necmwf number of elements of input grid

f 1D input forcing field

id index indicating the forcing variable

gridDesc Array describing the input grid

lb unpacked input bitmap

fret return code (0-indicates success)

31.1.7 interp_iv (Source File: retecmwf.F90)

INTERFACE:

```
subroutine interp_iv (n, iv,lis_gridDesc,necmwf,nlis,lb,f,lis_1d,mo,iret)
```

USES:

```
use lisdrv_module, only : lis
use ecmwfdomain_module, only : ecmwf_struct
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer :: iv
real :: lis_gridDesc(50)
integer :: necmwf
integer :: nlis
real :: f(necmwf)
real, dimension(nlis) :: lis_1d
integer :: mo
integer :: iret
```

DESCRIPTION:

This routine interpolates the input forcing to the LIS grid
The arguments are:

n index of the nest

iv field identifier

lis_gridDesc array description of the LIS grid

necmwf number of elements in the input grid

nlis number of elements in the LIS grid

f input data array to be interpolated

lis_1d interpolated array in the LIS grid

mo Number of points in the output grid

iret return code (0 indicates success)

The routines invoked are:

bilinear_interp (7.0.3)

spatially interpolate the forcing data using bilinear interpolation

31.1.8 time_interp_ecmwf (Source File: time_interp_ecmwf.F90)

REVISION HISTORY:

```
1 Oct 1999: Jared Entin; Initial code
25 Oct 1999: Jared Entin; Significant F90 Revision
11 Apr 2000: Brian Cosgrove; Fixed name construction error
              in Subroutine ETA6HRFFILE
27 Apr 2000: Brian Cosgrove; Added correction for use of old shortwave
              data with opposite sign convention from recent shortwave data.
              Added capability to use time averaged shortwave & longwave data
              Altered times which are passed into ZTERP--used to be GMT1
              and GMT2, now they are LDAS%ETATIME1 and LDAS%ETATIME2
30 Nov 2000: Jon Radakovich; Initial code based on geteta.f
17 Apr 2001: Jon Gottschalck; A few changes to allow model init.
13 Aug 2001: Urszula Jambor; Introduced missing data replacement.
5 Nov 2001: Urszula Jambor; Reset tiny negative SW values to zero.
```

INTERFACE:

```
subroutine time_interp_ecmwf(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use baseforcing_module, only : lisforc
use listime_mgr
use spmdMod
use ecmwfdomain_module, only : ecmwf_struct
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep. Downward shortwave radiation is interpolated using a zenith-angled based approach. Precipitation and longwave radiation are not temporally interpolated, and the previous 3 hourly value is used. All other variables are linearly interpolated between the 3 hourly blocks.

The routines invoked are:

time2date (5.4.21)

converts the time to a date format

tick (5.4.22)

advances or retracts time by the specified amount

zterp (5.26.5)

zenith-angle based interpolation

31.1.9 ecmwfforcing_finalize (Source File: ecmwfforcing_finalize.F90)

REVISION HISTORY:

25Oct2005; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine ecmwfforcing_finalize
```

USES:

```
use lisdrv_module, only : lis
use ecmwfdomain_module
```

DESCRIPTION:

Routine to cleanup allocated structures for ECMWF forcing.

32 GDAS

The Global Data Assimilation System (GDAS) is the global, operational weather forecast model developed at the Environmental Modeling Center (EMC) of NOAA/NCEP [6]. The radiation algorithms in GDAS include parameterizations for SW↓ and LW↓ radiation interactions between clouds and radiation. The shortwave radiation parameterization follows the scheme developed by [3] and the longwave radiation scheme is based on the work of [19]. GDAS forcing variables are produced on a quadratic T170 gaussian grid. LIS uses the 00, 03, and as needed, the 06 (hours after current) forecasts, which are produced at 6 hour intervals. From October 2002 onwards, the GDAS outputs are generated at a higher resolution T254 gaussian grid.

32.1 Fortran: Module Interface gdasdomain_module (Source File: gdasdomain_module.F90)

This module contains variables and data structures that are used for the implementation of the forcing data from the Global Data Assimilation System (GDAS) developed at the Environmental Modeling Center (EMC) of NOAA/NCEP. GDAS forcing variables are produced on a quadratic T170 gaussian grid. LIS uses the 00, 03, and as needed, the 06 forecasts, which are produced at 6 hour intervals. From October 2002 onwards, GDAS outputs are generated at a higher resolution T254 gaussian grid.

The implementation in LIS has the derived data type `gdas_struct` that includes the variables that specify the runtime options, and the weights and neighbor information to be used for spatial interpolation. They are described below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

nmif Number of forcing variables in the GDAS data

gdastime1 The nearest, previous 3 hour instance of the incoming data (as a real time).

gdastime2 The nearest, next 3 hour instance of the incoming data (as a real time).

griduptime1 The time to switch GDAS resolution to T170

griduptime2 The time to switch GDAS resolution to T254

gdasdir Directory containing the input data

elevfile File with the elevation definition for the input data.

mi Number of points in the input grid

rlat1 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

rlon1 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

n11,n121,n211,n221 Arrays containing the neighbor information of the input grid for each grid point in LIS, for bilinear interpolation.

w111,w121,w211,w221 Arrays containing the weights of the input grid for each grid point in LIS, for bilinear interpolation.

rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

n112,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: defineNativeGDAS      !defines the native resolution of
                                !the input data
```

PUBLIC TYPES:

```
public :: gdas_struct
```

32.1.1 defineNativeGDAS (Source File: gdasdomain_module.F90)

REVISION HISTORY:

11Dec2003: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine defineNativeGDAS()
```

USES:

```
use lisdrv_module, only: lis
use listime_mgr, only : date2time
```

DESCRIPTION:

Defines the native resolution of the input forcing for GDAS data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readgdasrcd (32.1.2)

reads the runtime options specified for GDAS data

date2time (5.4.20)

converts date to the real time format

bilinear_interp_input (7.0.2)

computes the neighbor, weights for bilinear interpolation

conserv_interp_input (7.0.4)

computes the neighbor, weights for conservative interpolation

read_gdas_elev (32.1.3)

reads the native elevation of the GDAS data to be used for topographic adjustments to the forcing

32.1.2 readgdascrd (Source File: readgdascrd.F90)

REVISION HISTORY:

11 Dec 2003; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine readgdascrd()
```

USES:

```
use lisdrv_module, only : lis, config_lis
use lis_logmod, only : logunit
use LIS_ConfigMod
use gdasdomain_module, only: gdas_struc
```

DESCRIPTION:

This routine reads the options specific to GDAS forcing from the LIS configuration file.
The routines invoked are:

LIS_ConfigGetAttribute (5.7.5)

read the specified attribute

LIS_ConfigFindLabel (5.7.8)

find the specified label to read the attribute

32.1.3 read_gdas_elev (Source File: read_gdas_elev.F90)

REVISION HISTORY:

17Dec2004; Sujay Kumar; Initial Specficaton

INTERFACE:

```
subroutine read_gdas_elev(n, phase)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use baseforcing_module, only :lisforc
use gdasdomain_module, only : gdas_struc
use lis_logmod, only : logunit
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: phase
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates GDAS model elevation to the LIS grid. The data will be used to perform any topographical adjustments to the forcing.

The arguments are:

n index of the nest

32.1.4 getgdas (Source File: getgdas.F90)

REVISION HISTORY:

1 Oct 1999: Jared Entin; Initial code
25 Oct 1999: Jared Entin; Significant F90 Revision
11 Apr 2000: Brian Cosgrove; Fixed name construction error
in Subroutine ETA6HFILE
27 Apr 2000: Brian Cosgrove; Added correction for use of old shortwave
data with opposite sign convention from recent shortwave data.
Added capability to use time averaged shortwave and longwave data.
Altered times which are passed into ZTERP--used to be GMT1 and GMT2,
now they are LDAS%ETATIME1 and LDAS%ETATIME2
11 May 2000: Brian Cosgrove; Added checks for SW values that are too high
due to zenith angle weighting...if too high, use linear weighting.
Also, if cos(zen) less than .1, then use linear weighting to
avoid computed values of SW that are too high.
18 May 2000: Brian Cosgrove; Corrected line of code in ETAEDASNAME which
assigned wrong year directory variable when constructing
EDAS filename
26 May 2000: Jared Entin; Changed numerical bound of the TRY variable
to fix a rollback problem.
5 June 2000: Brian Cosgrove; Fixed a problem with the correction of the negative
radiation sign convention. Prior to fix, was not correcting negative
values of -999.9...now it changes all negative values to positive ones.
18 Aug 2000: Brian Cosgrove; Fixed undefined value problem in check for negative
radiation values over land points.
08 Dec 2000: Urszula Jambor; Rewrote geteta.f in fortran90 to use GDAS in GLDAS
15 Mar 2001: Jon Gottschalck; Slight change to handle more forcing parameters at
time step 0.
09 Apr 2001: Urszula Jambor; Added capability of using DAAC forcing data every
6 hours, rather than every 3 hours.
30 May 2001: Urszula Jambor; Changed forcing used: T,q,u fields taken
from F00 & F03 files, radiation and precip. fields taken
from F06 & F03 (F03 fields are subtracted out from F06)

INTERFACE:

```
subroutine getgdas(n)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use baseforcing_module, only: lisforc
use listime_mgr
use gdasdomain_module, only : gdas_struc
use spmdMod
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates 3-hrly, GDAS forcing. The idea is to open either the 00 or 03 forecast file associated with the most recent GDAS assimilation (available every 6 hours). Precipitation rates and radiation fluxes will be taken from the F03 and F06 files, since averages are provided. At the beginning of a simulation, the code reads the most recent past data (nearest 3 hour interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. The strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day . The arguments are:

n index of the nest

The routines invoked are:

tick (5.4.22)
determines the GDAS data times

gdasfile (32.1.5)
Puts together appropriate file name for 3 hour intervals

gdasfilef06 (32.1.6)
Puts together appropriate file name for 6 hour intervals

retgdas (32.1.7)
Interpolates GDAS data to LIS grid

bilinear_interp_input (7.0.2)
computes the neighbor, weights for bilinear interpolation

conserv_interp_input (7.0.4)
computes the neighbor, weights for conservative interpolation

read_gdas_elev (32.1.3)
reads the native elevation of the GDAS data to be used for topographic adjustments to the forcing

32.1.5 gdasfile (Source File: getgdas.F90)

INTERFACE:

```
subroutine gdasfile( name, gdasdir, yr, mo, da, hr )  
  
    implicit none
```

ARGUMENTS:

```
character(len=80), intent(in)      :: gdasdir  
integer, intent(in)                :: yr, mo, da, hr  
character(len=80), intent (out)   :: name
```

DESCRIPTION:

This subroutine puts together GDAS file name for 3 hour file intervals
The arguments are:

gdasdir Name of the GDAS directory

yr year

mo month

da day of month

hr hour of day

name name of the timestamped GDAS file

32.1.6 gdasfilef06 (Source File: getgdas.F90)

INTERFACE:

```
subroutine gdasfileF06( name, gdasdir, yr, mo, da, hr )
```

USES:

```
use listime_mgr
```

```
implicit none
```

ARGUMENTS:

```
character(len=80) :: gdasdir
integer           :: yr, mo, da, hr
character(len=80) :: name
```

DESCRIPTION:

This subroutine puts together GDAS file name for 6 hour forecast files.

The arguments are:

gdasdir Name of the GDAS directory

yr year

mo month

da day of month

hr hour of day

name name of the timestamped GDAS file

32.1.7 retgdas (Source File: retgdas.F90)

REVISION HISTORY:

```
14 Dec 2000: Urszula Jambor; Rewrote geteta.f to use GDAS forcing in GLDAS
15 Mar 2001: Jon Gottschalck; Added additional parameters and octets in
              which to search in GRIB files
01 Jun 2001: Urszula Jambor; Added option to get forcing from different
              files (F00 instantaneous and F06 six hour means)
29 Jan 2003: Urszula Jambor; Rewrote code, uses GETGB call to replace
              ungribgdas. Interpolation now occurs in interp_gdas.
              Using GETGB avoids problems with the Oct2002 GDAS
              grid update.
12 Nov 2003: Matt Rodell; Check to make sure input file exists before
              opening and thereby creating a new, empty file.
14 Nov 2003: Matt Rodell; Ensure lubb varies in call to baopen
05 Feb 2004: James Geiger; Added GrADS-DODS Server functionality
```

INTERFACE:

```
subroutine retgdas( order, n, name, nameF06, F00flag,ferror,try )
```

USES:

```
use lisdrv_module, only : lis,lisdom
use listime_mgr
use baseforcing_module, only: lisforc
use gdasdomain_module, only : gdas_struc
use spmdMod
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: order
integer, intent(in)      :: n
character(len=80), intent(in) :: name, nameF06
integer, intent(in)      :: F00flag
integer, intent(out)     :: ferror
integer, intent(inout)    :: try
```

DESCRIPTION:

For the given time, reads parameters from GDAS data, transforms into 9 LIS forcing parameters and interpolates to the LIS domain.

The arguments are:

order flag indicating which data to be read (order=1, read the previous 3 hourly instance, order=2, read the next 3 hourly instance)

n index of the nest

name name of the 3 hour GDAS forecast file

nameF06 name of the 6 hour GDAS forecast file

ferror flag to indicate success of the call (=0 indicates success)

try index of the tries (in case of missing data)

The routines invoked are:

interp_gdas (32.1.8)

spatially interpolates a GDAS variable

32.1.8 interp_gdas (Source File: retgdas.F90)

INTERFACE:

```
subroutine interp_gdas(n, kpds,ngdas,f,lb,lis_gds,nc,nr, &
                      varfield)
```

USES:

```

use lisdrv_module, only : lis
use gdasdomain_module, only : gdas_struct

implicit none

```

ARGUMENTS:

```

integer, intent(in) :: n
integer, intent(in) :: kpds(200)
integer, intent(in) :: ngdas
real, intent(out)   :: f(ngdas)
logical*1           :: lb(ngdas)
real                :: lis_gds(50)
integer, intent(in) :: nc
integer, intent(in) :: nr
real, intent(out)   :: varfield(nc,nr)

```

DESCRIPTION:

This subroutine interpolates a given GDAS field to the LIS grid. The arguments are:

n index of the nest

kpds grib decoding array

ngdas number of elements in the input grid

f input data array to be interpolated

lb input bitmap

lis_gds array description of the LIS grid

nc number of columns (in the east-west dimension) in the LIS grid

nr number of rows (in the north-south dimension) in the LIS grid

varfield output interpolated field

The routines invoked are:

bilinear_interp (7.0.3)

spatially interpolate the forcing data using bilinear interpolation

conserv_interp (7.0.5)

spatially interpolate the forcing data using conservative interpolation

32.1.9 time_interp_gdas (Source File: time_interp_gdas.F90)

REVISION HISTORY:

```

1 Oct 1999: Jared Entin; Initial code
25 Oct 1999: Jared Entin; Significant F90 Revision
11 Apr 2000: Brian Cosgrove; Fixed name construction error
              in Subroutine ETA6HFILE
27 Apr 2000: Brian Cosgrove; Added correction for use of old shortwave
              data with opposite sign convention from recent shortwave data.
              Added capability to use time averaged shortwave & longwave data

```

Altered times which are passed into ZTERP--used to be GMT1
 and GMT2, now they are LDAS%ETATIME1 and LDAS%ETATIME2
 30 Nov 2000: Jon Radakovich; Initial code based on geteta.f
 17 Apr 2001: Jon Gottschalck; A few changes to allow model init.
 13 Aug 2001: Urszula Jambor; Introduced missing data replacement.
 5 Nov 2001: Urszula Jambor; Reset tiny negative SW values to zero.

INTERFACE:

```
subroutine time_interp_gdas(n)
```

USES:

```
use lisdrv_module, only :lis, lisdom
use baseforcing_module, only: lisforc
use listime_mgr
use spmdMod
use gdasdomain_module, only : gdas_struct
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep. Downward shortwave radiation is interpolated using a zenith-angled based approach. Precipitation and longwave radiation are not temporally interpolated, and the previous 3 hourly value is used. All other variables are linearly interpolated between the 3 hourly blocks.

The arguments are:

n index of the nest

The routines invoked are:

time2date (5.4.21)

converts the time to a date format

tick (5.4.22)

advances or retracts time by the specified amount

zterp (5.26.5)

zenith-angle based interpolation

32.2 Fortran: Module Interface gdasforcing_finalize (Source File: gdasforcing_finalize.F90)

REVISION HISTORY:

25Oct2005; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine gdasforcing_finalize
```

USES:

```
use lisdrv_module, only : lis
use gdasdomain_module
```

DESCRIPTION:

Routine to cleanup allocated structures for GDAS forcing.

33 GEOS

This section describes the implementation of the forcing data available from the Goddard Earth Observing System (GEOS) data assimilation system produced by the NASA GSFC's Global Modeling and Assimilation office (GMAO).

33.1 Fortran: Module Interface `geosdomain_module` (Source File: `geosdomain_module.F90`)

This module contains variables and data structures that are used for the implementation of the GEOS forcing data. The data is global 1 degree dataset in latlon projection, and at 3 hourly intervals. The derived data type `geos_struct` includes the variables that specify the runtime options, and the weights and neighbor information to be used for spatial interpolation. They are described below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

nmif Number of forcing variables in the ECMWF data

geostime1 The nearest, previous 3 hour instance of the incoming data (as a real time).

geostime2 The nearest, next 3 hour instance of the incoming data (as a real time).

griduptime The time to switch GEOS resolution

geosdir Directory containing the input data

mi Number of points in the input grid

rlat1 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

rlon1 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

n111,n121,n211,n221 Arrays containing the neighbor information of the input grid for each grid point in LIS, for bilinear interpolation.

w111,w121,w211,w221 Arrays containing the weights of the input grid for each grid point in LIS, for bilinear interpolation.

rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

n122,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: defineNativeGEOS      !defines the native resolution of
                                !the input data
```

PUBLIC TYPES:

```
public :: geos_struct
```

33.1.1 defineNativeGEOS (Source File: geosdomain_module.F90)

REVISION HISTORY:

11Dec2003: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine defineNativeGEOS
```

USES:

```
use lisdrv_module, only: lis
use listime_mgr, only : date2time
implicit none
```

DESCRIPTION:

Defines the native resolution of the input forcing for GEOS data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readgeoscrd (33.1.2)

reads the runtime options specified for GEOS data

bilinear_interp_input (7.0.2)

computes the neighbor, weights for bilinear interpolation

conserv_interp_input (7.0.4)

computes the neighbor, weights for conservative interpolation

33.1.2 readgeoscrd (Source File: readgeoscrd.F90)

REVISION HISTORY:

11 Dec 2003; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine readgeoscrd()
```

USES:

```
use lisdrv_module, only : lis, config_lis
use LIS_ConfigMod
use lis_logmod, only : logunit
use geosdomain_module, only : geos_struc
```

DESCRIPTION:

This routine reads the options specific to GDAS forcing from the LIS configuration file.
The routines invoked are:

LIS_ConfigGetAttribute (5.7.5)

read the specified attribute

LIS_ConfigFindLabel (5.7.8)

find the specified label to read the attribute

33.1.3 getgeos (Source File: getgeos.F90)

REVISION HISTORY:

```
1 Oct 1999: Jared Entin; Initial code
25 Oct 1999: Jared Entin; Significant F90 Revision
11 Apr 2000: Brian Cosgrove; Fixed name construction error
             in Subroutine ETA6HFILE
27 Apr 2000: Brian Cosgrove; Added correction for use of old shortwave
             data with opposite sign convention from recent shortwave data.
             Added capability to use time averaged shortwave & longwave data
             Altered times which are passed into ZTERP--used to be GMT1
             and GMT2, now they are LDAS%ETATIME1 and LDAS%ETATIME2
30 Nov 2000: Jon Radakovich; Initial code based on geteta.f
17 Apr 2001: Jon Gottschalck; A few changes to allow model init.
13 Aug 2001: Urszula Jambor; Introduced missing data replacement.
5 Nov 2001: Urszula Jambor; Reset tiny negative SW values to zero.
```

INTERFACE:

```
subroutine getgeos(n)
```

USES:

```
use lisdrv_module, only : lis
use lis_logmod,only : logunit
use listime_mgr
use spmdMod
use baseforcing_module, only: lisforc
use geosdomain_module, only : geos_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates 3-hrly, 1 degree GEOS forcing. At the beginning of a simulation, the code reads the most recent past data (nearest 3 hour interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. The strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day . The arguments are:

n index of the nest

The routines invoked are:

tick (5.4.22)
call to advance or retract time

geosfile (33.1.4)
Puts together appropriate file name for 3 hour intervals

readgeos (33.1.5)
call to read the GEOS data and perform spatial interpolation

bilinear_interp_input (7.0.2)
computes the neighbor, weights for bilinear interpolation

conserv_interp_input (7.0.4)
computes the neighbor, weights for conservative interpolation

33.1.4 geosfile (Source File: getgeos.F90)

INTERFACE:

```
subroutine geosfile(name,geosdir,yr,mo,da,hr,ncold)
```

```
    implicit none
```

ARGUMENTS:

```
character*40, intent(in) :: geosdir  
integer, intent(in)      :: yr,mo,da,hr,ncold  
character*80, intent(out) :: name
```

DESCRIPTION:

This subroutine puts together GEOS file name for 3 hour file intervals
The arguments are:

geosdir Name of the GEOS directory

yr year

mo month

da day of month

hr hour of day

ncold Number of columns (used as to determine the GEOS resolution)

name name of the timestamped GEOS file

33.1.5 readgeos (Source File: readgeos.F90)

REVISION HISTORY:

```
1 Oct 1999: Jared Entin; Initial code
15 Oct 1999: Paul Houser; Significant F90 Revision
11 Apr 2000: Brian Cosgrove; Added read statements for forcing interpolation
17 Apr 2001: Jon Gottschalck; Added code to perform initialization of
               Mosaic with GEOS forcing and new intp. scheme
14 Aug 2001: Urszula Jambor; Added ferror flag as a routine argument
07 Dec 2001: Urszula Jambor; Began used of LDAS$$LDAS_GRIDDESC array
```

INTERFACE:

```
subroutine readgeos(order,n, name,tscount,ferror)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use spmdMod
use lis_logmod,only : logunit
use baseforcing_module, only: lisforc
use geosdomain_module, only : geos_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: n
character*80, intent(in) :: name
integer, intent(in)      :: order, tscount
integer, intent(out)     :: ferror
```

DESCRIPTION:

For the given time, reads parameters from GEOS data, transforms into 9 LIS forcing parameters and interpolates to the LIS domain.

GEOS FORCING VARIABLES (unless noted, fields are 3-hr upstream averaged):

1. T 2m Temperature interpolated to 2 metres [K]
2. q 2m Instantaneous specific humidity interpolated to 2 metres[kg/kg]
3. radswg Downward shortwave flux at the ground [W/m²]
4. lwgdn Downward longwave radiation at the ground [W/m²]
5. u 10m Instantaneous zonal wind interpolated to 10 metres [m/s]
6. v 10m Instantaneous meridional wind interpolated to 10 metres[m/s]
7. ps Instantaneous Surface Pressure [Pa]
8. preacc Total precipitation [mm/s]
9. precon Convective precipitation [mm/s]
10. albedo Surface albedo (0-1)

The arguments are:

order flag indicating which data to be read (order=1, read the previous 3 hourly instance, order=2, read the next 3 hourly instance)

n index of the nest

name name of the 3 hour GDAS forecast file

tscount time step count

ferror return error code (0 indicates success)

The routines invoked are:

bilinear_interp (7.0.3)

spatially interpolate the forcing data using bilinear interpolation

conserv_interp (7.0.5)

spatially interpolate the forcing data using conservative interpolation

33.1.6 time_interp_geos (Source File: time_interp_geos.F90)

REVISION HISTORY:

```
1 Oct 1999: Jared Entin; Initial code
25 Oct 1999: Jared Entin; Significant F90 Revision
11 Apr 2000: Brian Cosgrove; Fixed name construction error
              in Subroutine ETA6HFILE
27 Apr 2000: Brian Cosgrove; Added correction for use of old shortwave
              data with opposite sign convention from recent shortwave data.
              Added capability to use time averaged shortwave & longwave data
              Altered times which are passed into ZTERP--used to be GMT1
              and GMT2, now they are LDAS%ETATIME1 and LDAS%ETATIME2
30 Nov 2000: Jon Radakovich; Initial code based on geteta.f
17 Apr 2001: Jon Gottschalck; A few changes to allow model init.
13 Aug 2001: Urszula Jambor; Introduced missing data replacement.
5 Nov 2001: Urszula Jambor; Reset tiny negative SW values to zero.
```

INTERFACE:

```
subroutine time_interp_geos(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use baseforcing_module, only : lisforc
use listime_mngr
use lis_logmod,only : logunit
use spmdMod
use geosdomain_module, only :geos_struct

implicit none
```

ARGUMENTS:

```
integer, intent(in):: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep. Downward shortwave radiation is interpolated using a zenith-angled based approach. Precipitation and longwave radiation are not temporally interpolated, and the previous 3 hourly value is used. All other variables are linearly interpolated between the 3 hourly blocks.

The routines invoked are:

time2date (5.4.21)

converts the time to a date format

tick (5.4.22)

advances or retracts time by the specified amount

zterp (5.26.5)

zenith-angle based interpolation

33.1.7 geosforcing_finalize (Source File: geosforcing_finalize.F90)

REVISION HISTORY:

25Oct2005; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine geosforcing_finalize
```

USES:

```
use lisdrv_module, only : lis
use geosdomain_module
```

DESCRIPTION:

Routine to cleanup geos forcing related memory allocations.

34 GSWP

This section describes the implementation of the forcing data used in the Global Soil Wetness Project.

34.1 Fortran: Module Interface gswpdomain_module (Source File: gswpdomain_module.F90)

This module contains variables and data structures that are used for the implementation of the forcing data from the Global Soil Wetness Project (GSWP). GSWP forcing variables are produced on a latlon 1degree grid at 3 hour intervals.

The implemenatation in LIS has the derived data type **gswp_struct** that includes the variables that specify the runtime options They are described below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

nmif Number of forcing variables in the GSWP data

gswptime1 The nearest, previous 3 hour instance of the incoming data (as a real time).

gswptime2 The nearest, next 3 hour instance of the incoming data (as a real time).

tair Directory containing the 2m air temperature data

qair Directory containing the 2m specific humidity data

psurf Directory containing the surface pressure data

wind Directory containing the wind data
rainf Directory containing the total precipitation data
snowf Directory containing the total snowfall data
swdown Directory containing the downward shortwave radiation data
swdown Directory containing the downward longwave radiation data
mi Number of points in the input grid

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: defineNativeGSWP      !defines the native resolution of
                           !the input data
```

PUBLIC TYPES:

```
public :: gswp_struct
```

34.1.1 defineNativeGSWP (Source File: gswpdomain_module.F90)

REVISION HISTORY:

11Dec2003: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine defineNativeGSWP()
```

USES:

```
use lisdrv_module, only: lis
use listime_mgr, only : date2time
```

DESCRIPTION:

Defines the native resolution of the input forcing for GSWP data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readgswpcrd (34.1.2)
reads the runtime options specified for GSWP data

34.1.2 readgswpcrd (Source File: readgswpcrd.F90)

REVISION HISTORY:

11 Dec 2003; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine readgswpcrd()
```

USES:

```
use lis_logmod, only : logunit
use lisdrv_module, only : lis, config_lis
use gswpdomain_module, only : gswp_struc
use LIS_ConfigMod
```

DESCRIPTION:

This routine reads the options specific to GSWP forcing from the LIS configuration file.
The routines invoked are:

LIS_ConfigGetAttribute (5.7.5)

read the specified attribute

LIS_ConfigFindLabel (5.7.8)

find the specified label to read the attribute

34.1.3 getgswp (Source File: getgswp.F90)

REVISION HISTORY:

20Feb2004; Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine getgswp(n)
```

USES:

```
use lisdrv_module, only : lis
use listime_mngr
use spmdMod
use baseforcing_module, only: lisforc
use gswpdomain_module, only : gswp_struc
use lis_logmod, only : logunit,lis_log_msg
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates 3-hrly, GSWP forcing. At the beginning of a simulation, the code reads the most recent past data (nearest 3 hour interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. The strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day . The arguments are:

n index of the nest

The routines invoked are:

tick (5.4.22)
determines the GSWP data times

readgswp (34.1.4)
Interpolates GSWP data to LIS grid

34.1.4 readgswp (Source File: **readgswp.F90**)

REVISION HISTORY:

20Feb2004; Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine readgswp(order,n, yr,mo,da,hr,mn,ss)
```

USES:

```
#if ( defined USE_NETCDF )
  use netcdf
#endif
  use lisdrv_module, only : lis,lisdom
  use baseforcing_module, only:lisforc
  use gswp_module, only : getgswp_timeindex
  use gswpdomain_module, only : gswp_struc
  use lis_logmod, only : logunit
```

```
implicit none
```

ARGUMENTS:

```
  integer, intent(in) :: n
```

DESCRIPTION:

For the given time, reads parameters from the GSWP files, transforms into 9 LIS forcing parameters and interpolates to the LIS domain.

The arguments are:

order flag indicating which data to be read (order=1, read the previous 3 hourly instance, order=2, read the next 3 hourly instance)

n index of the nest

yr current year

mo current month

da current day of the month

hr current hour of day

mn current minute

ss current second

The routines invoked are:

getgswp_timeindex (5.22.2)
computes the GSWP time index

34.1.5 time_interp_gswp (Source File: time_interp_gswp.F90)

REVISION HISTORY:

20Feb2004; Sujay Kumar : Initial Specification

INTERFACE:

```
subroutine time_interp_gswp(n)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use baseforcing_module, only : lisforc
use listime_mngr
use spmdMod
use gswpdomain_module, only :gswp_struc
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep. Downward shortwave radiation is interpolated using a zenith-angled based approach. Precipitation and longwave radiation are not temporally interpolated, and the previous 3 hourly value is used. All other variables are linearly interpolated between the 3 hourly blocks.

The routines invoked are:

time2date (5.4.21)

converts the time to a date format

zterp (5.26.5)

zenith-angle based interpolation

34.1.6 finterp (Source File: finterp.F90)

REVISION HISTORY:

2002/08/16 15:55:18 guo

INTERFACE:

```
subroutine finterp(ip,trp_flag,val0,val1,val2,val3,madtt,istep,vout)
```

USES:

```
use lis_logmod, only : logunit

implicit none

integer,intent(in)      :: ip
```

```

integer,intent(in)      :: istep
character(len=1),intent(in) :: trp_flag
real,intent(in)          :: val0
real,intent(in)          :: val1
real,intent(in)          :: val2
real,intent(in)          :: val3
integer,intent(in)      :: madtt
real                     :: vout

```

DESCRIPTION:

Introduced a new interpolation type "P" which specifies a temporal disaggregation of interpolated precipitation to improve partitioning of infiltration/runoff when forcing data interval is large compared to the typical length of a convective (or total) rain event.

** This routine interpolates atmospheric adtt seconds period forcing data to dtt seconds valtrp time-step for one adtt interval. Interpolation is performed based on the value of 'trp_flag':

"L" or "l" = value represents average over interval ending at current time
 "N" or "n" = value represents average over interval beginning at current time
 "C" or "c" = value represents average over interval centered on current time
 "I" or "i" = instantaneous value at current time (linear interpolation)
 "P" or "p" = PDF disaggregation applied in time (for precip)
 "0" (zero) = no interpolation, centered on current time
 Otherwise = no interpolation, applied beginning at current time

* NOTE:

* For "L", "N", and "C" to conserve the mean over the interval after
 * interpolation, 'madtt' MUST BE A MULTIPLE OF 2!

The arguments are:

ip grid point in question
trp_flag type of interpolation
val0 last forcing value (for cases N and C)
val1 current forcing value (all cases)
val2 next forcing value (for all cases but default)
val3 ueber-next forcing value (for all cases C and L)
madtt number of valtrp timesteps in a forcing timestep
vout output forcing value

34.2 Fortran: Module Interface gswpforcing_finalize (Source File: gswpforcing_finalize.F90)

REVISION HISTORY:

25Oct2005; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine gswpforcing_finalize
```

USES:

```
use gswpdomain_module
```

DESCRIPTION:

Routine to cleanup allocated structures for GSWP forcing.

35 NLDAS

The atmospheric forcing used in the North American Land Data Assimilation System (NLDAS) [5] features products at an hourly 0.125° spatial resolution, over the continental United States. The SW^\downarrow is generated from a $1/2^\circ$ product derived at the University of Maryland from NOAA's Geostationary Operational Environmental Satellites (GOES). The LW^\downarrow is derived from 3 hourly NCEP Eta Data Assimilation System (EDAS) output fields [17], and from 3 hourly and 6 hourly Eta mesoscale model forecast fields when EDAS data is unavailable.

35.1 Fortran: Module Interface `nldasdomain_module` (Source File: `nldasdomain_module.F90`)

This module contains variables and data structures that are used for the implementation of the forcing data used in the North American Land Data Assimilation System (NLDAS; Cosgrove et al.(2003)). The variables are produced at 0.125 degree spatial resolution, and at hourly intervals.

Cosgrove, B. Real-time and retrospective forcing in the North American Land Data Assimilation System (NLDAS) project. *Journal of Geophysical Research*, 108 (D22), 8842, DOI:10.1029/2002JD003118

The implemenatation in LIS has the derived data type `nldas_struct` that includes the variables that specify the runtime options, and the weights and neighbor information to be used for spatial interpolation. They are described below:

ncold Number of columns (along the east west dimension) for the input data

nrcld Number of rows (along the north south dimension) for the input data

nldastime1 The nearest, previous hourly instance of the incoming data (as a real time).

nldastime2 The nearest, next hourly instance of the incoming data (as a real time).

nldasdir Directory containing the input data

elevfile File with the elevation definition for the input data.

mi Number of points in the input grid

rlat1 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

rlon1 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for bilinear interpolation)

n111,n121,n211,n221 Arrays containing the neighbor information of the input grid for each grid point in LIS, for bilinear interpolation.

w111,w121,w211,w221 Arrays containing the weights of the input grid for each grid point in LIS, for bilinear interpolation.

rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

n122,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: defineNativeNLDAS      !defines the native resolution of
                           !the input data
```

PUBLIC TYPES:

```
public :: nldas_struct
```

35.1.1 defineNativeNLDAS (Source File: nldasdomain.module.F90)

REVISION HISTORY:

02Feb2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine defineNativeNLDAS()
```

USES:

```
use lisdrv_module, only: lis
use listime_mgr, only : date2time
```

DESCRIPTION:

Defines the native resolution of the input forcing for NLDAS data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readnldascrd (35.1.2)

reads the runtime options specified for NLDAS data

date2time (5.4.20)
converts date to the real time format

bilinear_interp_input (7.0.2)
computes the neighbor, weights for bilinear interpolation

conserv_interp_input (7.0.4)
computes the neighbor, weights for conservative interpolation

read_nldas_elev (35.1.3)
reads the native elevation of the NLDAS data to be used for topographic adjustments to the forcing

35.1.2 readnldascrd (Source File: readnldascrd.F90)

REVISION HISTORY:

02Feb2004; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine readnldascrd()
```

USES:

```
use lisdrv_module, only : lis
use nldasdomain_module, only : nldas_struct
use spmdMod, only : masterproc
use lis_logmod, only : logunit
use lisdrv_module, only : config_lis
use LIS_ConfigMod
```

DESCRIPTION:

This routine reads the options specific to NLDAS forcing from the LIS configuration file.
The routines invoked are:

LIS_ConfigGetAttribute (5.7.5)
read the specified attribute

LIS_ConfigFindLabel (5.7.8)
find the specified label to read the attribute

35.1.3 read_nldas_elev (Source File: read_nldas_elev.F90)

REVISION HISTORY:

17Dec2004; Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine read_nldas_elev(n)
```

USES:

```

use lisdrv_module, only : lis,lisdom
use baseforcing_module, only : lisforc
use nldasdomain_module, only : nldas_struc
use lis_logmod, only : logunit
use map_utils

implicit none

```

ARGUMENTS:

```

integer, intent(in) :: n

```

DESCRIPTION:

Opens, reads, and interpolates NLDAS model elevation to the LIS grid. The data will be used to perform any topographical adjustments to the forcing.

The arguments are:

n index of the nest

The routines invoked are:

ij_to_latlon (7.1.9)

computes the lat lon values in LIS grid projection

35.1.4 getnldas (Source File: getnldas.F90)

REVISION HISTORY:

```

1 Oct 1999: Jared Entin; Initial code
15 Oct 1999: Paul Houser; Significant F90 Revision
20 Dec 1999: Paul Houser; Allow for Eta Data Overwrite by NCEP data
27 Apr 2000: Brian Cosgrove; Turned zenith angle weighting back on.
             Changed times supplied to ZTERP from GMT1 and GMT2 to
             LDAS%NLDASTIME1 and LDAS%NLDASTIME2
4 May 2000: Added 15 minutes to GMT1 and GMT2 to accurately
            reflect the valid times of the NCEP radiation data
18 Aug 2000: Brian Cosgrove; Fixed error in date calculations so that
            3600 second (1hr) timestep may be used.
            Added code to make any calculated radiation forcing
            values undefined if both ncep time 1 and ncep time 2
            radiation values are undefined
27 Feb 2001: Brian Cosgrove; Added CZM into call for ZTERP subroutine
07 Mar 2001: Brian Cosgrove; Added code to allow for use of NASA-LDAS data
04 Sep 2001: Brian Cosgrove; Changed tempgmt1,tempgmt2 to real to match
            tick.f call, changed file name construction.
21 Aug 2002: Brian Cosgrove; Removed code that adjusted for 15 minute
            offset in radiation fields supplied by NOAA or NASA
            NLDAS realtime or retrospective standard forcing files.
            This offset no longer exists as it is now dealt with
            during forcing file creation through zenith angle correction.
            The need for this fix was just discovered...unfortunately
            simulations before the date of this fix using
            this fortran subroutine have incorrectly shifted
            radiation data.
02Feb 2004 : Sujay Kumar ; Initial Version in LIS

```

INTERFACE:

```
subroutine getnldas(n)
```

USES:

```
use spmdMod, only : masterproc
use lisdrv_module, only : lis, lisdom
use baseforcing_module, only : lisforc
use nldasdomain_module, only : nldas_struc
use listime_mgr, only : tick
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates hourly, NLDAS forcing. At the beginning of a simulation, the code reads the most recent past data (nearest hourly interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. The strategy for missing data is to go backwards up to 10 days to get forcing at the same time of day . The arguments are:

n index of the nest

The routines invoked are:

tick (5.4.22)

determines the NLDAS data times

nldasfile (35.1.5)

Puts together appropriate timestamped filename

retnldas (35.1.6)

Interpolates NLDAS data to LIS grid

bilinear_interp_input (7.0.2)

computes the neighbor, weights for bilinear interpolation

conserv_interp_input (7.0.4)

computes the neighbor, weights for conservative interpolation

read_nldas_elev (35.1.3)

reads the native elevation of the NLDAS data to be used for topographic adjustments to the forcing

35.1.5 nldasfile (Source File: **getnldas.F90**)

REVISION HISTORY:

```
1 Oct 1999: Jared Entin; Initial code
15 Oct 1999: Paul Houser; Significant F90 Revision
04 Sep 2001: Brian Cosgrove; Use of NASA data enabled, updated
              reading of data directory structure to read new format
```

INTERFACE:

```
subroutine nldasfile(name,nldasdir,yr,mo,da,hr)  
    implicit none
```

ARGUMENTS:

```
character*80, intent(out):: name  
character*40, intent(in) :: nldasdir  
integer, intent(in)      :: yr,mo,da,hr
```

DESCRIPTION:

This subroutine puts together NLDAS file name for 3 hour file intervals
The arguments are:

nldasdir Name of the NLDAS directory

yr year

mo month

da day of month

hr hour of day

name name of the timestamped NLDAS file

35.1.6 retnldas (Source File: retnldas.F90)

REVISION HISTORY:

```
1 Oct 1999: Jared Entin; Initial code  
15 Oct 1999: Paul Houser; Significant F90 Revision  
11 Apr 2000: Brian Cosgrove; changed code to use Forcing Mask (With inland  
water filled in). Deleted unused variables.  
27 Apr 2000: Brian Cosgrove; changed code to use the original  
mask again since that is the  
mask which NCEP has already applied to the forcing data  
by the time NASA gets it.....not possible to use the  
expanded NASA forcing mask  
1 May 2000: Brian Cosgrove; changed code so that if parameter 11 (sw)  
is not found in hourly ncep data, it will just use  
edas-based shortwave from the hourly ncep files  
20 Jun 2000: Brian Cosgrove; changed code so that it uses LDAS%UDEF and  
not a hard-wired undefined value of -999.9 and -999.0  
18 Aug 2000: Brian Cosgrove; changed code so that FMASK and not MASK  
is used when ungridding. NCEP data already has a mask applied  
to it and so may not be able to supply forcing data to  
all LDAS land forcing points. In areas where LDAS  
forcing mask states that land exists, but where NCEP forcing  
data is non-existent, assign undefined value to forcing data.  
22 Aug 2000: Brian Cosgrove; Altered code for US/Mexico/Canada Mask  
05 Sep 2001: Brian Cosgrove; Removed dirnom and infile variables, changed  
call to ungrindncep to match removal. Added code to make use  
of precip weighting mask  
02 Feb 2004: Sujay Kumar; Initial Specification in LIS
```

INTERFACE:

```
subroutine retnldas(order,n, name,ferror)
```

USES:

```
use lisdrv_module, only : lis,lisdom
use nldasdomain_module, only : nldas_struct
use baseforcing_module, only : lisforc
use lis_logmod, only : logunit
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in)      :: order
integer, intent(in)      :: n
character*80, intent(in) :: name
integer, intent(out)     :: ferror
```

DESCRIPTION:

For the given time, reads parameters from NLDAS data, transforms into 9 LIS forcing parameters and interpolates to the LIS domain.

The arguments are:

order flag indicating which data to be read (order=1, read the previous hourly instance, order=2, read the next hourly instance)

n index of the nest

name name of the 3 hour GDAS forecast file

ferror flag to indicate success of the call (=0 indicates success)

The routines invoked are:

interp_nldas (35.1.7)
spatially interpolates a NLDAS variable

35.1.7 interp_nldas (Source File: retnldas.F90)

INTERFACE:

```
subroutine interp_nldas(n, kpds, nldas,f,lb,lis_gds,nc,nr, &
varfield)
```

USES:

```
use lisdrv_module, only : lis
use nldasdomain_module, only : nldas_struct

implicit none
```

ARGUMENTS:

```

integer, intent(in) :: n
integer, intent(in) :: kpds(200)
integer, intent(in) :: nldas
real, intent(out) :: f(nldas)
logical*1, intent(in) :: lb(nldas)
real, intent(in) :: lis_gds(50)
integer, intent(in) :: nc
integer, intent(in) :: nr
real, intent(inout) :: varfield(nc,nr)

```

DESCRIPTION:

This subroutine interpolates a given NLDAS field to the LIS grid. The arguments are:

n index of the nest

kpds grib decoding array

nldas number of elements in the input grid

f input data array to be interpolated

lb input bitmap

lis_gds array description of the LIS grid

nc number of columns (in the east-west dimension) in the LIS grid

nr number of rows (in the north-south dimension) in the LIS grid

varfield output interpolated field

The routines invoked are:

bilinear_interp (7.0.3)

spatially interpolate the forcing data using bilinear interpolation

conserv_interp (7.0.5)

spatially interpolate the forcing data using conservative interpolation

35.1.8 time_interp_nldas (Source File: time_interp_nldas.F90)

REVISION HISTORY:

02Feb2004: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine time_interp_nldas(n)
```

USES:

```

use lisdrv_module, only : lis,lisdom
use baseforcing_module, only : lisforc
use nldasdomain_module, only : nldas_struct
use listime_mngr, only : tick, time2date
use spmdMod, only : iam
use lis_logmod, only : logunit

implicit none

```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep. Downward shortwave radiation is interpolated using a zenith-angled based approach. Precipitation and longwave radiation are not temporally interpolated, and the previous 3 hourly value is used. All other variables are linearly interpolated between the 3 hourly blocks.

The routines invoked are:

time2date (5.4.21)

converts the time to a date format

tick (5.4.22)

advances or retracts time by the specified amount

zterp (5.26.5)

zenith-angle based interpolation

35.2 Fortran: Module Interface `nldasforcing_finalize` (Source File: `nldasforcing_finalize.F90`)

REVISION HISTORY:

25Oct2005; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine nldasforcing_finalize
```

USES:

```
use lisdrv_module, only : lis
use nldasdomain_module
```

DESCRIPTION:

Routine to cleanup nldas forcing related memory allocations.

Part XI

Supplemental forcing analyses in LIS

36 Overview

This section contains some examples of the supplemental forcing implementations in LIS. These implementations, by definition overwrite the corresponding base forcing analyses.

37 CMAP

This section describes the implementation of the precipitation forcing from the Climate Prediction Center's (CPC) Merged Analysis of Precipitation (CMAP)

37.1 Fortran: Module Interface `cmapdomain_module` (Source File: `cmapdomain_module.F90`)

This module contains variables and data structures that are used for the implementation of the precipitation data from the Climate Prediction Center (CPC)'s Merged Analysis of Precipitation (CMAP). CMAP merges gauge measurements and satellite estimates including GPI, OPI, SSM/I to produce a global 2.5 degree pentad precipitation analysis. The 6-hourly product obtained by disaggregating CMAP using the GDAS forcing is used in this implementation.

The implementation in LIS has the derived data type `cmap_struct` that includes the variables to specify the runtime options, and the weights and neighbor information for spatial interpolation

They are described below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

cmapdir Directory containing the input data

cmaptime The nearest, hourly instance of the incoming data (as a real time).

griduptime1 The time to switch the input resolution to T170

mi Number of points in the input grid

rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

n112,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: defineNativeCMAP      !defines the native resolution of
                           !the input data
```

PUBLIC TYPES:

```
public :: cmap_struct
```

37.1.1 defineNativeCMAP (Source File: cmapdomain_module.F90)

REVISION HISTORY:

11Dec2003: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine defineNativeCMAP()
```

USES:

```
use lisdrv_module, only: lis
use listime_mgr, only : date2time
```

DESCRIPTION:

Defines the native resolution of the input forcing for CMAP data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V

The routines invoked are:

readcmapcrd (37.1.2)

reads the runtime options specified for CMAP data

date2time (5.4.20)

converts date to the real time format

conserv_interp_input (7.0.4)

computes the neighbor, weights for conservative interpolation

37.1.2 readcmapcrd (Source File: readcmapcrd.F90)

REVISION HISTORY:

11 Dec 2003; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine readcmapcrd()
```

USES:

```
use cmapdomain_module, only : cmap_struct
use lisdrv_module, only : config_lis,lis
use LIS_ConfigMod
use lis_logmod, only : logunit
```

DESCRIPTION:

This routine reads the options specific to CMAP forcing from the LIS configuration file.
The routines invoked are:

LIS_ConfigGetAttribute (5.7.5)

read the specified attribute

LIS_ConfigFindLabel (5.7.8)

find the specified label to read the attribute

37.1.3 getcmap (Source File: getcmap.F90)

REVISION HISTORY:

```
17 Jul 2001: Jon Gottschalck; Initial code  
10 Oct 2001: Jon Gottschalck; Modified to adjust convective precip  
              using a ratio of the model convective / total ratio
```

INTERFACE:

```
subroutine getcmap(n)
```

USES:

```
use lisdrv_module, only : lis  
use listime_mngr  
use lis_logmod, only : logunit  
use cmapdomain_module, only : cmap_struct  
  
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates 6-hrly, CMAP forcing. At the beginning of a simulation, the code reads the most recent past data (nearest 6 hour interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. . The arguments are:

n index of the nest

The routines invoked are:

tick (5.4.22)
determines the CMAP data times

cmapfile (37.1.4)
Puts together appropriate file name for 6 hour intervals

glbprecip_cmap (37.1.5)
Interpolates CMAP data to LIS grid

conserv_interp_input (7.0.4)
computes the neighbor, weights for conservative interpolation

37.1.4 cmapfile (Source File: getcmap.F90)

INTERFACE:

```
subroutine cmapfile( name, cmapdir, yr, mo, da, hr)
```

```
implicit none
```

ARGUMENTS:

```
character(len=80) :: name, cmapdir
integer :: yr, mo, da, hr
```

DESCRIPTION:

This subroutine puts together CMAP file name for 6 hour file intervals
The arguments are:

cmapdir Name of the CMAP directory

yr year

mo month

da day of month

hr hour of day

name name of the timestamped CMAP file

37.1.5 glbprecip_cmap (Source File: glbprecip_cmap.F90)

REVISION HISTORY:

17 Jul 2001: Jon Gottschalck; Initial code

04 Feb 2002: Jon Gottschalck; Added necessary code to use global precip
observations with domain 3 (2x2.5)

INTERFACE:

```
subroutine glbprecip_cmap( n, fname, ferror_cmap, filehr)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use lis_logmod, only : logunit
use cmapdomain_module, only : cmap_struct
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
character(len=80) :: fname
integer :: ferror_cmap
integer :: filehr
```

DESCRIPTION:

For the given time, reads parameters from CMAP data and interpolates to the LIS domain.
The arguments are:

n index of the nest

fname name of the 6 hour CMAP file

ferror_cmap flag to indicate success of the call (=0 indicates success)

filehr current file hour

The routines invoked are:

interp_cmap (37.1.6)

spatially interpolates the CMAP data

37.1.6 interp_cmap (Source File: interp_cmap.F90)

INTERFACE:

```
subroutine interp_cmap(n, kpds,ncmap,f,lb,lis_gds,nc,nr, &
varfield)
```

USES:

```
use cmapdomain_module, only : cmap_struc
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer :: nc
integer :: nr
integer :: ncmap
integer :: kpds(200)
real :: lis_gds(50)
real :: f(ncmap)
logical*1 :: lb(ncmap)
real, dimension(nc,nr) :: varfield
```

DESCRIPTION:

This subroutine interpolates a given CMAP field to the LIS grid. The arguments are:

n index of the nest

kpds grib decoding array

ncmap number of elements in the input grid

f input data array to be interpolated

lb input bitmap

lis_gds array description of the LIS grid

nc number of columns (in the east-west dimension) in the LIS grid

nr number of rows (in the north-south dimension) in the LIS grid

varfield output interpolated field

The routines invoked are:

conserv_interp (7.0.5)

spatially interpolate the forcing data using conservative interpolation

37.1.7 time_interp_cmap (Source File: time_interp_cmap.F90)

INTERFACE:

```
subroutine time_interp_cmap(n)
```

USES:

```
use lisdrv_module, only: lis, lisdom
use cmapdomain_module, only : cmap_struct

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep.
The arguments are:

n index of the nest

38 Huffman

This section describes the implementation of the precipitation product derived from the TRMM satellite precipitation analysis generated by George Huffman at NASA GSFC.

38.1 Fortran: Module Interface huffdomain_module (Source File: huffdomain_module.F90)

This module contains variables and data structures that are used for the implementation of the precipitation product derived from the TRMM real-time multi-satellite precipitation analysis from NASA GSFC (Huffman et al.2003)

Huffman, G.J.,R.F. Adler, E.F.Stocker, D.T.Bolvin, and E.J.Nelkin (2003): Analysis of TRMM 3-hourly multi-satellite precipitation estimates computed in real and post-real time. Combined preprints CD-ROM, 83rd AMS Annual Meeting, Paper P4.11 in 12th Conference on Sat. Meteor. and Oceanog. 9-13 Feb 2003, Long Beach CA. 6pp.

The implementation in LIS has the derived data type **huff_struct** that includes the variables to specify the runtime options, and the weights and neighbor information for spatial interpolation

They are described below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

huffdir Directory containing the input data

hufftime The nearest, hourly instance of the incoming data (as a real time).

griduptime1 The time to switch the input resolution to T170

mi Number of points in the input grid

rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

n112,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

USES:

```
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: defineNativeHUFF      !defines the native resolution of  
                                !the input data
```

PUBLIC TYPES:

```
public :: huff_struct
```

38.1.1 defineNativeHuff (Source File: huffdomain_module.F90)

REVISION HISTORY:

11Dec2003: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine defineNativeHuff()
```

USES:

```
use lisdrv_module, only: lis  
use listime_mgr, only : date2time
```

DESCRIPTION:

Defines the native resolution of the input forcing for Huffman data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readhuffcrd (38.1.2)

reads the runtime options specified for Huffman data

date2time (5.4.20)

converts date to the real time format

conserv_interp_input (7.0.4)

computes the neighbor, weights for conservative interpolation

38.1.2 readhuffcrd (Source File: readhuffcrd.F90)

REVISION HISTORY:

11 Dec 2003; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine readhuffcrd()
```

USES:

```
use huffdomain_module, only : huff_struct
use lisdrv_module, only : lis,config_lis
use LIS_ConfigMod
use lis_logmod, only : logunit
```

DESCRIPTION:

This routine reads the options specific to CMAP forcing from the LIS configuration file.
The routines invoked are:

LIS_ConfigGetAttribute (5.7.5)

read the specified attribute

LIS_ConfigFindLabel (5.7.8)

find the specified label to read the attribute

38.1.3 gethuff (Source File: gethuff.F90)

REVISION HISTORY:

17 Jul 2001: Jon Gottschalck; Initial code

10 Oct 2001: Jon Gottschalck; Modified to adjust convective precip
using a ratio of the model convective / total ratio

INTERFACE:

```
subroutine gethuff(n)
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr
use lis_logmod, only : logunit
use spmdMod,only : masterproc
use huffdomain_module, only : huff_struct

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates 3-hrly, Huffman forcing. At the beginning of a simulation, the code reads the most recent past data (nearest 3 hour interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. . The arguments are:

n index of the nest

The routines invoked are:

tick (5.4.22)
determines the Huffman data times

hufffile (38.1.4)
Puts together appropriate file name for 6 hour intervals

glbprecip_huff (38.1.5)
Interpolates Huffman data to LIS grid

conserv_interp_input (7.0.4)
computes the neighbor, weights for conservative interpolation

38.1.4 hufffile (Source File: gethuff.F90)

INTERFACE:

```
subroutine hufffile( name, huffdir, yr, mo, da, hr)
implicit none
```

ARGUMENTS:

```
character(len=80) :: name, huffdir
integer           :: yr, mo, da, hr
```

DESCRIPTION:

This subroutine puts together Huffman file name for 3 hour file intervals
The arguments are:

huffdir Name of the Huffman data directory

yr year

mo month

da day of month

hr hour of day

name name of the timestamped Huffman file

38.1.5 glbprecip_huff (Source File: glbprecip_huff.F90)

REVISION HISTORY:

17 Jul 2001: Jon Gottschalck; Initial code
04 Feb 2002: Jon Gottschalck; Added necessary code to use global precip observations with domain 3 (2x2.5)

INTERFACE:

```
subroutine glbprecip_huff (n, name_huff, ferror_huff )
```

USES:

```
use lisdrv_module, only : lis, lisdom
use huffdomain_module, only : huff_struc
use spmdMod,only : masterproc
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

For the given time, reads parameters from Huffman data and interpolates to the LIS domain.
The arguments are:

n index of the nest

name_huff name of the 3 hour Huffman forecast file

ferror_huff flag to indicate success of the call (=0 indicates success)

The routines invoked are:

interp_huff (38.1.6)

spatially interpolates the Huffman data

38.1.6 interp_huff (Source File: interp_huff.F90)

INTERFACE:

```
subroutine interp_huff(n, nx, ny, finput, lis_gds, nc, nr, varfield)
```

USES:

```
use huffdomain_module, only : huff_struc

implicit none
```

ARGUMENTS:

```

integer, intent(in) :: n
integer :: nx
integer :: ny
integer :: nc
integer :: nr
real :: lis_gds(50)
real, dimension(nx,ny) :: finput

```

DESCRIPTION:

This subroutine interpolates a given Huffman field to the LIS grid. The arguments are:

n index of the nest
nx number of columns (in the east-west dimension) in the Huffman grid
ny number of rows (in the north-south dimension) in the Huffman grid
finput input data array to be interpolated
lis_gds array description of the LIS grid
nc number of columns (in the east-west dimension) in the LIS grid
nr number of rows (in the north-south dimension) in the LIS grid
varfield output interpolated field

The routines invoked are:

conserv_interp (7.0.5)
 spatially interpolate the forcing data using conservative interpolation

38.1.7 time_interp_huff (Source File: time_interp_huff.F90)

INTERFACE:

```
subroutine time_interp_huff(n)
```

USES:

```

use lisdrv_module, only:lis,lisdom
use huffdomain_module, only : huff_struc

implicit none

```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep.
 The arguments are:

n index of the nest

39 CMORPH

This section describes the implementation of the precipitation product from CPC's MORPHing technique, known as CMORPH.

39.1 Fortran: Module Interface `cmordomain_module` (Source File: `cmordomain_module.F90`)

REVISION HISTORY:

05Jan2006: Yudong Tian; start from Jon G.'s code for older versions of LIS.

DESCRIPTION:

This module contains variables and data structures that are used for the implementation of the precipitation data from the Climate Prediction Center (CPC)'s MORPHing technique (CMORPH). CMORPH products are produced as global fields from 60 N-60S at 8km resolution at 30 minute intervals.

The implementation in LIS has the derived data type `cmor_struct` that includes the variables to specify the runtime options, and the weights and neighbor information for spatial interpolation. They are described below:

ncold Number of columns (along the east west dimension) for the input data

nrold Number of rows (along the north south dimension) for the input data

cmordir Directory containing the input data

cmortime The nearest, hourly instance of the incoming data (as a real time).

griduptime1 The time to switch the input resolution to T170

mi Number of points in the input grid

rlat2 Array containing the latitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

rlon2 Array containing the longitudes of the input grid for each corresponding grid point in LIS (to be used for conservative interpolation)

n112,n122,n212,n222 Arrays containing the neighbor information of the input grid for each grid point in LIS, for conservative interpolation.

w112,w122,w212,w222 Arrays containing the weights of the input grid for each grid point in LIS, for conservative interpolation.

USES:

`implicit none`

PUBLIC MEMBER FUNCTIONS:

```
public :: defineNativeCMORPH      !defines the native resolution of
          !the input data
```

PUBLIC TYPES:

```
public :: cmor_struct
```

39.1.1 defineNativeCMORPH (Source File: cmordomain_module.F90)

REVISION HISTORY:

11Dec2003: Sujay Kumar; Initial Specification
06Jan2006: Yudong Tian; modification for LISv4.2

INTERFACE:

```
subroutine defineNativeCMORPH
```

USES:

```
use lisdrv_module, only: lis
use listime_mngr, only : date2time

implicit none

real :: gridDesci(50)
integer :: updoj, yr1,mo1,da1,hr1,mn1,ss1
real :: upgmt
integer :: n
```

DESCRIPTION:

Defines the native resolution of the input forcing for CMORPH data. The grid description arrays are based on the decoding schemes used by NCEP and followed in the LIS interpolation schemes V
The routines invoked are:

readcmorcrd (39.1.2)
reads the runtime options specified for CMORPH data

date2time (5.4.20)
converts date to the real time format

conserv_interp_input (7.0.4)
computes the neighbor, weights for conservative interpolation

39.1.2 readcmorcrd (Source File: readcmorcrd.F90)

REVISION HISTORY:

11 Dec 2003; Sujay Kumar, Initial Code
29 Dec 2003; Luis Gustavo, adapted for CMORPH
06 Jan 2005; Yudong Tian, modified for LISv4.2

INTERFACE:

```
subroutine readcmorcrd()
```

USES:

```
use cmordomain_module, only : cmor_struct
use lisdrv_module, only : config_lis, lis
use LIS_ConfigMod
use lis_logmod, only : logunit
```

DESCRIPTION:

This routine reads the options specific to CMORPH forcing from the LIS configuration file.
The routines invoked are:

LIS_ConfigGetAttribute (5.7.5)

read the specified attribute

LIS_ConfigFindLabel (5.7.8)

find the specified label to read the attribute

39.1.3 getcmor (Source File: getcmor.F90)

REVISION HISTORY:

17 Jul 2001: Jon Gottschalck; Initial code

10 Oct 2001: Jon Gottschalck; Modified to adjust convective precip
using a ratio of the model convective / total ratio

29 Dec 2003: Luis Goncalves; Added CMORPH global observed precip data sources

06 Jan 2005: Yudong Tian; Modified for LISv4.2

INTERFACE:

```
subroutine getcmor(n)
```

USES:

```
use lisdrv_module, only : lis
use listime_mngr
use spmdMod,only : masterproc
use cmordomain_module, only :cmor_struct
use lis_logmod, only : logunit
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Opens, reads, and interpolates 30 minute CMORPH forcing. At the beginning of a simulation, the code reads the most recent past data (nearest 30min interval), and the nearest future data. These two datasets are used to temporally interpolate the data to the current model timestep. . The arguments are:

n index of the nest

The routines invoked are:

tick (5.4.22)

determines the CMORPH data times

cmorfile (39.1.4)

Puts together appropriate file name for 6 hour intervals

glbprecip_cmor (39.1.5)

Interpolates CMORPH data to LIS grid

conserv_interp_input (7.0.4)

computes the neighbor, weights for conservative interpolation

39.1.4 cmorfile (Source File: getcmor.F90)

INTERFACE:

```
subroutine cmorfile( name, cmordir, yr, mo, da, hr)  
    implicit none
```

ARGUMENTS:

```
character(len=100) :: name, cmordir  
integer :: yr, mo, da, hr
```

DESCRIPTION:

This subroutine puts together CMORPH file name for 30min file intervals
The arguments are:

cmordir Name of the CMORPH directory

yr year

mo month

da day of month

hr hour of day

name name of the timestamped CMORPH file

39.1.5 glbprecip_cmor (Source File: glbprecip_cmor.F90)

REVISION HISTORY:

```
17 Jul 2001: Jon Gottschalck; Initial code  
04 Feb 2002: Jon Gottschalck; Added necessary code to use global precip  
             observations with domain 3 (2x2.5)  
29 Dec 2003: Luis Goncalves; Added code to use CMORPH precip data  
06 Jan 2006: Yudong Tian; modified for LISv4.2
```

INTERFACE:

```
subroutine glbprecip_cmor (n, name_cmor, ferror_cmor, iflg )
```

USES:

```
use lisdrv_module, only : lis, lisdom  
use cmordomain_module, only : cmor_struc  
use spmdMod,only : masterproc  
use lis_logmod, only : logunit  
  
    implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

For the given time, reads parameters from CMORPH data and interpolates to the LIS domain.
The arguments are:

n index of the nest

name_cmor name of the CMORPH file

ferror_cmor flag to indicate success of the call (=0 indicates success)

iflg flag indicating which 1/2 hour to read

The routines invoked are:

interp_cmor (39.1.6)

spatially interpolates the CMORPH data

39.1.6 interp_cmor (Source File: interp_cmor.F90)

INTERFACE:

```
subroutine interp_cmor(n, nx, ny, finput, lis_gds, nc, nr, varfield)
```

USES:

```
use cmordomain_module, only : cmor_struct
```

```
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer :: nx
integer :: ny
integer :: nc
integer :: nr
real, dimension(nx,ny) :: finput
real, dimension(nc,nr) :: varfield
```

DESCRIPTION:

This subroutine interpolates a given CMORPH field to the LIS grid. The arguments are:

n index of the nest

nx number of columns (in the east-west dimension) in the CMORPH grid

ny number of rows (in the north-south dimension) in the CMORPH grid

finput input data array to be interpolated

lis_gds array description of the LIS grid

nc number of columns (in the east-west dimension) in the LIS grid

nr number of rows (in the north-south dimension) in the LIS grid

varfield output interpolated field

The routines invoked are:

conserv_interp (7.0.5)

spatially interpolate the forcing data using conservative interpolation

39.1.7 time_interp_cmor (Source File: time_interp_cmor.F90)

INTERFACE:

```
subroutine time_interp_cmor(n)
```

USES:

```
use lisdrv_module, only:lis,lisdom
use cmordomain_module, only : cmor_struc
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

Temporally interpolates the forcing data to the current model timestep.
The arguments are:

n index of the nest

Part XII

Land Surface Models in LIS

40 Land Surface Models in LIS

This section contains the description of interface implementations specific to different land surface models. The implementations include Noah LSM from NCEP, the community land model (CLM), Mosaic and catchment from NASA GSFC, and VIC from Princeton University. It is to be noted that the LIS reference manual documents only the interface hooks into LIS and it does not document the land surface physics routines. Please contact the developers of the individual model for same.

41 Noah land surface model version 2.6

This section describes the interface implementations for the Noah land surface model. The community Noah Land Surface Model is a stand-alone, uncoupled, 1-D column model freely available at the National Centers for Environmental Prediction (NCEP; [2]). The name is an acronym representing the various developers of the model (N: NCEP; O: Oregon State University, Dept. of Atmospheric Sciences; A: Air Force (both AFWA and AFRL - formerly AFGL, PL); and H: Hydrologic Research Lab - NWS (now Office of Hydrologic Development – OHD)). Noah can be executed in either coupled or uncoupled mode. It has been coupled with the operational NCEP mesoscale Eta model [7] and its companion Eta Data Assimilation System (EDAS) [17], and the NCEP Global Forecast System (GFS) and its companion Global Data Assimilation System (GDAS). When Noah is executed in uncoupled mode, near-surface atmospheric forcing data (e.g., precipitation, radiation, wind speed, temperature, humidity) is required as input. Noah simulates soil moisture (both liquid and frozen), soil temperature, skin temperature, snowpack depth, snowpack water equivalent, canopy water content, and the energy flux and water flux terms of the surface energy balance and surface water balance. The model applies finite-difference spatial discretization methods and a Crank-Nicholson time-integration scheme to numerically integrate the governing equations of the physical processes of the soil vegetation-snowpack medium, including the surface energy balance equation, Richards' [15] equation for soil hydraulics, the diffusion equation for soil heat transfer, the energy-mass balance equation for the snowpack, and the Jarvis [9] equation for the conductance of canopy transpiration.

41.1 Fortran: Module Interface noah_module (Source File: noah_module.F90)

The code in this file provides a description of the data structure containing the noah 1-d variables. The variables specified in the data structure include:

soiltype soil type (integer index)

nroot Number of root layers, depends on the vegetation type

rsmin Minimum canopy resistance (s/m)

rg1 Solar radiation term of canopy resistance function

hs vapor pressure deficit term of canopy resistance function

snup threshold snow depth (in water equivalent m) that implies 100% snow cover

z0 roughness length

lai leaf area index

vegip interpolated vegetation fraction

vegmp1 month 1 greenness fraction value

vegmp2 month 2 greenness fraction value

albsf1 date 1 snow free albedo value

albsf2 date 2 snow free albedo value

albsf interpolated snow free albedo value
smcmax maximum soil moisture content (porosity)
psisat saturated matric potential
dksat saturated hydraulic conductivity
bexp b parameter value
quartz quartz fraction
smcwlt wilting point (volumetric)
smcref reference soil moisture (onset of soil moisture stress in transpiration, volumetric)
dwsat saturated soil diffusivity
mxsnoalb maximum albedo expected over deep snow
tempbot bottom boundary temperature
tempbot1 month 1 bottom boundary temperature
tempbot2 month 2 bottom boundary temperature
t1 skin temperature (K)
cmc canopy water content
snowh actual snow depth (m)
sneqv snow water equivalent (m)
stc soil temperature for different layers
smc soil moisture for different layers
sh2o liquid-only soil moisture for different layers
ch heat/moisture exchange coefficient
cm momentum exchange coefficient
forcing array of meteorological forcing
vegt vegetation type of tile
swnet net shortwave radiation (W/m²)
lwnet net longwave radiation (W/m²)
qle latent heat flux (W/m²)
qh sensible heat flux (W/m²)
qg ground heat flux (W/m²)
snowf snowfall (kg/m²s)
rainf rainfall (kg/m²s)
evap evapotranspiration (kg/m²s)
qs surface runoff (kg/m²s)

qsb subsurface runoff (kg/m²s)
qsm snow melt (kg/m²s)
avgsurft average surface temperature (K)
albedo surface albedo (-)
swe snow water equivalent (kg/m²)
soilmoist soil moisture for different layers (kg/m²)
soilwet total column soil wetness (-)
ecanop interception evaporation (kg/m²s)
canopint total canopy water storage (kg/m²s)
rootmoist root zone soil moisture (kg/m²)
soilm_prev soil moisture from the previous model output
swe_prev snow water equivalent from the previous model output

REVISION HISTORY:

28 Apr 2002: K. Arsenault added NOAH LSM 2.5 code to LDAS.
14 Nov 2002: Sujay Kumar Optimized version for LIS

41.2 Fortran: Module Interface noah_varder (Source File: noah_varder.F90)

This module provides the definition of derived data type used to control the operation of Noah LSM. It also provides the entry method for the initialization of Noah-specific variables. The derived data type **noah_struct** includes the variables that specify the runtime options and other control variables as described below:

rfile name of the noah restart file
vfile name of the static vegetation parameter table
sfile name of the soil parameter table
pfile name of the general parameter table
count variable to keep track of the number of timesteps before an output
noahopen variable to keep track of opened files
soilscheme type of soil mapping scheme usd (1-zobler,2-statsgo)
numout number of output times
nslay number of soil layers
nvegp number of static vegetation parameters in the table
nsoilp number of soil parameters in the table
nstxts number of soil texture classes in the classification scheme

npr number of prognostic state variables used in data assimilation
varid NETCDF ids for variables (used for netcdf output)
tbotAlarmTime alarm time to keep track of TBOT data reading frequency
tbotInterval data interval of the TBOT data
initsm initial soil moisture for a cold start run
initTemp initial soil temperature for a cold start run
outInterval output writing interval
rstInterval restart writing interval
obsz height of meteorological observations
lyrthk thickness of soil layers
noah,noahbk noah LSM specific variables

REVISION HISTORY:

Apr 2003; Sujay Kumar, Initial Code

USES:

```
use noah_module  
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: noah_varder_ini
```

PUBLIC TYPES:

```
public :: noah_struc
```

41.2.1 noah_varder_ini (Source File: noah_varder.F90)

INTERFACE:

```
subroutine noah_varder_ini()
```

USES:

```
use lisdrv_module, only : lis
```

DESCRIPTION:

This routine creates the datatypes and allocates memory for noah-specific variables. It also invokes the routine to read the runtime specific options for noah from the configuration file.
The routines invoked are:

readnoahcrd (41.2.2)
reads the runtime options for Noah LSM

41.2.2 readnoahcrd (Source File: readnoahcrd.F90)

REVISION HISTORY:

14 Oct 2003; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine readnoahcrd()
```

USES:

```
use lisdrv_module, only : lis
use spmdMod, only : masterproc
use noah_varder, only : noah_struct
use lis_logmod, only : logunit
use lisdrv_module, only : config_lis
use LIS_ConfigMod
```

DESCRIPTION:

This routine reads the options specific to Noah LSM from the LIS configuration file.
The routines invoked are:

LIS_ConfigGetAttribute (5.7.5)

read the specified attribute

LIS_ConfigFindLabel (5.7.8)

find the specified label to read the attribute

41.2.3 noah_setvegparms (Source File: noah_setvegparms.F90)

REVISION HISTORY:

28 Apr 2002: Kristi Arsenault; Added NOAH LSM, Initial Code

13 Oct 2003: Sujay Kumar; Domain independent modifications

25 Jul 2005: Sujay Kumar; Removed the dependency to SIB classes

04 Oct 2005: Matthew Garcia; additions for Noah Unified

04 Oct 2005: Matthew Garcia; additions for Distributed Noah Router

INTERFACE:

```
subroutine noah_setvegparms
```

USES:

```
use lisdrv_module, only : lis,lisdom
use noah_varder
use spmdMod, only : iam,masterproc
```

DESCRIPTION:

This subroutine retrieves NOAH vegetation parameters. The current implementation uses a table-based lookup based on SIB classes to initialize the following parameters

```
nroot - number of root zones
rsmin - minimum canopy resistance
rgl - solar radiation term in canopy resistance function
hs - vapor pressure deficit term
snup - threshold snow depth
z0 - roughness length
lai - leaf area index
```

41.2.4 noah_setup (Source File: noah_setup.F90)

REVISION HISTORY:

4 Nov. 1999: Paul Houser; Initial Code
28 Apr. 2002: K. Arsenault; Modified to NOAH LSM 2.5 code to LDAS

INTERFACE:

```
subroutine noah_setup()
```

USES:

```
use lisdrv_module, only: lis
use noah_varder
use spmdMod, only : masterproc, npes
use listime_mgr, only : setMonthlyAlarm
```

DESCRIPTION:

This routine is the entry point to set up the parameters required for Noah LSM. These include the soils, greenness, albedo, bottom temperature and the initialization of state variables in Noah.

The routines invoked are:

noah_setvegparms (41.2.3)

initializes the vegetation-related parameters in Noah

noah_settbot (41.2.6)

initializes the bottom temperature fields

noah_setsoils (41.2.5)

initializes the soil parameters

noah_coldstart (41.2.7)

initializes the noah state variables

setMonthlyAlarm (5.4.12)

sets a monthly alarm if a climatology of bottom temperature is being read

41.2.5 noah_setsoils (Source File: noah_setsoils.F90)

REVISION HISTORY:

28 Apr 2002: Kristi Arsenault; Added NOAH LSM, Initial Code
13 Oct 2003: Sujay Kumar; Domain independent modifications
15 Sep 2005: Matthew Garcia; update for ARMS project (Windows capability)
04 Oct 2005: Matthew Garcia; additions for Noah Unified

INTERFACE:

```
subroutine noah_setsoils
```

USES:

```
use lisdrv_module, only : lis, lisdom
use noah_varder
use spmdMod, only : iam,masterproc
use soils_module, only : lis_soils, soils_finalize
use lis_logmod, only : logunit
```

DESCRIPTION:

This subroutine sets the soil parameters in Noah. Noah uses a lookup table based on the input soil texture to derive these parameters.

The routines invoked are:

soiltype (5.26.3)

method to derive the soil texture type from the sand, silt, and clay fractions.

41.2.6 noah_settbot (Source File: **noah_settbot.F90**)

REVISION HISTORY:

28 Apr 2002: Kristi Arsenault; Added NOAH LSM, Initial Code
13 Oct 2003: Sujay Kumar; Domain independent modifications

INTERFACE:

```
subroutine noah_settbot
```

USES:

```
use lisdrv_module, only : lis, lisdom
use noah_varder
use spmdMod, only : iam
use listime_mgr, only : isMonthlyAlarmRinging, computeMonthlyWeights
```

DESCRIPTION:

This subroutine retrieves bottom boundary temperature for Noah LSM. Currently a static data is being used. If a climatology is available, it can be used by setting the appropriate frequency of the climatology

The routines invoked are:

readtbot (5.39.2)

retrieves the bottom temperature data

41.2.7 noah_coldstart (Source File: noah_coldstart.F90)

INTERFACE:

```
subroutine noah_coldstart()
```

USES:

```
use lisdrv_module, only: lis
use noah_varder
use listime_mgr
use spmdMod, only: iam,masterproc
use lis_logmod, only : logunit
```

DESCRIPTION:

This routine initializes the noah state variables with some predefined values uniformly for the entire domain. These initial values will be overwritten by the values read from the supplied Noah model restart file.

41.2.8 noah_dynsetup (Source File: noah_dynsetup.F90)

REVISION HISTORY:

28 Jan 2002: Sujay Kumar, Initial Specification

INTERFACE:

```
subroutine noah_dynsetup(n)
```

DESCRIPTION:

This routine sets up the time-dependent variables in Noah. Currently this routine is a placeholder

41.2.9 noah_f2t (Source File: noah_f2t.F90)

REVISION HISTORY:

15 Oct 1999: Paul Houser; Initial Code

INTERFACE:

```
subroutine noah_f2t(n, t, forcing)
```

USES:

```
use lisdrv_module , only : lis
use noah_varder

implicit none

ARGUMENTS:

integer, intent(in) :: n
real, intent(in)    :: forcing(lis%nf)
integer, intent(in) :: t
```

DESCRIPTION:

This routine transfers the LIS provided forcing onto the Noah model tiles.
The arguments are:

n index of the nest
t index of the tile
forcing input forcing array from LIS

41.2.10 noah_main (Source File: noah_main.F90)

INTERFACE:

```
subroutine noah_main(n)
```

USES:

```
use lisdrv_module, only : lis, lisdom
use gfrac_module, only : lis_gfrac
use albedo_module, only : lis_alb
use noah_varder
use lis_logmod, only : logunit

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This is the entry point for calling the Noah LSM physics. This routine calls the **SFLX** routine for Noah that performs the land surface computations, to solve for water and energy equations. For documentation of the **SFLX** and other noah routines, please see the "NOAH LSM USERS GUIDE" maintained at the public server at NOAA NCEP.

The arguments are:

n index of the nest

41.2.11 noah_output (Source File: noah_output.F90)

INTERFACE:

```
subroutine noah_output(n)
```

USES:

```
#if ( defined USE_NETCDF )
  use netcdf
#endif
use lisdrv_module, only : lis,metadata_output
use grid_module
```

```

use noah_varder
use spmdMod
use lis_fileIOMod, only : create_output_directory, &
    create_output_filename,  &
    create_stats_filename

implicit none

integer, intent(in) :: n

```

DESCRIPTION:

This subroutines sets up methods to write model output from Noah LSM A timestamped filename and directory are created first. The routine supports model output in three data formats: Binary, Grib and NETCDF.

The arguments are:

n index of the nest

The routines invoked are:

create_output_directory (5.23.7)

creates a time stamped output directory

create_output_filename (5.23.8)

creates a time stamped output filename

create_stats_filename (5.23.10)

creates a stats filename

noah_binout (41.2.12)

routine to write Noah model output in binary format

noah_gribout (41.2.13)

routine to write Noah model output in grib format

noah_netcdfout (41.2.14)

routine to write Noah model output in netcdf format

noah_totint (41.2.15)

resets the time averaged variables

41.2.12 noah_binout (Source File: noah_binout.F90)

REVISION HISTORY:

02 Dec 2003; Sujay Kumar, Initial Version

INTERFACE:

```
subroutine noah_binout(ftn,ftn_stats,n)
```

USES:

```
use lisdrv_module, only : lis,metadata_output
use drv_output_mod, only : drv_writevar_bin
use noah_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn,ftn_stats
```

DESCRIPTION:

This routine writes the specified noah variables into a binary output file.
The arguments are:

n index of the nest

ftn unit number for the model output file

ftn_stats unit number for the stats file

The routines invoked are:

drv_writevar_bin (5.21.2)
routine to write a variable to a binary output file

41.2.13 noah_gribout (Source File: noah_gribout.F90)

REVISION HISTORY:

02 Dec 2003; Sujay Kumar, Initial Version

INTERFACE:

```
subroutine noah_gribout(ftn,ftn_stats,n)
```

USES:

```
use lisdrv_module, only : lis
use drv_output_mod, only : drv_writevar_grib
use noah_varder
use spmdMod, only : masterproc
use listime_mgr, only : tick

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn,ftn_stats
```

DESCRIPTION:

This routine writes the specified noah variables into a grib output file.
The arguments are:

n index of the nest

ftn unit number for the model output file

ftn_stats unit number for the stats file

The routines invoked are:

drv_writevar_grib (5.21.5)

routine to write a variable to a grib output file

41.2.14 noah_ncdfout (Source File: noah_ncdfout.F90)

REVISION HISTORY:

02 Dec 2003; Sujay Kumar, Initial Version

INTERFACE:

```
subroutine noah_ncdfout(check, ftn,n)
```

USES:

```
#if (USE_NETCDF)
  use netcdf
#endif
  use lisdrv_module, only : lis
  use drv_output_mod, only : drv_writevar_ncdf
  use noah_varder
  use lis_logmod, only : logunit

  implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn
logical, intent(in) :: check
```

DESCRIPTION:

This routine writes the specified noah variables into a netcdf output file.
The arguments are:

n index of the nest

ftn unit number for the model output file

check flag to determine whether to write the header information

The routines invoked are:

drv_writevar_ncdf (5.21.6)

routine to write a variable to a netcdf output file

41.2.15 noah_totinit (Source File: noah_totinit.F90)

REVISION HISTORY:

14 Jun 2002 Sujay Kumar Initial Specification

INTERFACE:

```
subroutine noah_totinit(n)
```

USES:

```
use noah_varder
use lisdrv_module, only : lis

implicit none

integer, intent(in) :: n
```

DESCRIPTION:

This routine resets the time-averaged Noah variables after a model output. The routine is called after a model output call.

41.2.16 noah_writerst (Source File: noah_writerst.F90)

REVISION HISTORY:

```
1 Oct 1999: Jared Entin; Initial code
15 Oct 1999: Paul Houser; Significant F90 Revision
05 Sep 2001: Brian Cosgrove; Modified code to use Dag Lohmann's NOAA
              initial conditions if necessary. This is controlled with
              local variable NOAAIC. Normally set to 0 in this subroutine
              but set to 1 if want to use Dag's NOAA IC's. Changed output
              directory structure, and commented out if-then check so that
              directory is always made.
28 Apr 2002: Kristi Arsenault; Added NOAH LSM into LDAS
28 May 2002: Kristi Arsenault; For STARTCODE=4, corrected SNEQV values
              and put SMC, SH20, STC limit for GDAS and GEOS forcing.
14 Jun 2003: Sujay Kumar , Separated the write restart from the original
              code
```

INTERFACE:

```
subroutine noah_writerst(n)
```

USES:

```
use lisdrv_module, only : lis
use listime_mgr
use noah_varder
use lis_logmod, only : logunit
use lis_fileIOMod, only : create_output_directory, &
                        create_restart_filename
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This program writes restart files for Noah. This includes all relevant water/energy storage and tile information

The routines invoked are:

create_output_directory (5.23.7)

creates a timestamped directory for the restart files

create_restart_filename (5.23.9)

generates a timestamped restart filename

noah_dump_restart (41.2.17)

writes the noah variables into the restart file reads a variable from the restart file

41.2.17 noah_dump_restart (Source File: noah_writerst.F90)

REVISION HISTORY:

24 Aug 2004: James Geiger, Initial Specification

INTERFACE:

```
subroutine noah_dump_restart(n, ftn)
```

USES:

```
use lisdrv_module, only : lis
use noah_varder
use listime_mgr
use drv_output_mod, only : drv_writevar_restart

implicit none

integer, intent(in) :: ftn
integer, intent(in) :: n
```

DESCRIPTION:

This routine gathers the necessary restart variables and performs the actual write statements to create the restart files.

The arguments are:

n index of the nest

ftn unit number for the restart file

The following is the list of variables written in the Noah restart file:

```
nc,nr,nch      - grid and tile space dimensions
t1              - noah skin temperature
cmc             - noah canopy moisture storage
snowh           - noah snow depth
```

sneqv	- noah snow water equivalent
stc	- noah soil temperature (for each layer)
smc	- noah soil moisture (for each layer)
sh2o	- noah liquid only soil moisture (for each layer)
ch	- noah heat/moisture exchange coefficient
cm	- noah momentum exchange coefficient

The routines invoked are:

drv_writevar_restart (5.21.3)

writes a variable to the restart file

41.2.18 noahrst (Source File: noahrst.F90)

REVISION HISTORY:

1 Oct 1999: Jared Entin; Initial code
 15 Oct 1999: Paul Houser; Significant F90 Revision
 05 Sep 2001: Brian Cosgrove; Modified code to use Dag Lohmann's NOAA initial conditions if necessary. This is controlled with local variable NOAAIC. Normally set to 0 in this subroutine but set to 1 if want to use Dag's NOAA IC's. Changed output directory structure, and commented out if-then check so that directory is always made.
 28 Apr 2002: Kristi Arsenault; Added NOAH LSM into LDAS
 28 May 2002: Kristi Arsenault; For STARTCODE=4, corrected SNEQV values and put SMC, SH2O, STC limit for GDAS and GEOS forcing.

INTERFACE:

subroutine noahrst

USES:

```
use spmdMod, only : iam,masterproc
use lisdrv_module, only : lis
use noah_varder
use listime_mgr
use drv_output_mod, only : drv_readvar_restart
use lis_logmod, only : logunit
```

DESCRIPTION:

This program reads restart files for Noah. This includes all relevant water/energy storages and tile information. The following is the list of variables specified in the Noah restart file:

nc,nr,nch	- grid and tile space dimensions
t1	- noah skin temperature
cmc	- noah canopy moisture storage
snowh	- noah snow depth
sneqv	- noah snow water equivalent
stc	- noah soil temperature (for each layer)
smc	- noah soil moisture (for each layer)
sh2o	- noah liquid only soil moisture (for each layer)
ch	- noah heat/moisture exchange coefficient
cm	- noah momentum exchange coefficient

The routines invoked are:

drv_readvar_restart (5.21.4)
reads a variable from the restart file

41.2.19 noah_finalize (Source File: noah_finalize.F90)

REVISION HISTORY:

04 Nov 1999: Paul Houser; Initial Code
28 Apr 2002: K. Arsenault; Modified to NOAH LSM 2.5 code to LDAS
04 Oct 2005: Matthew Garcia; additions for Noah Unified

INTERFACE:

`subroutine noah_finalize()`

USES:

```
use lisdrv_module, only : lis
use noah_varder
```

DESCRIPTION:

This routine cleans up the allocated memory structures in Noah

42 Community Land Model version 2.0

This section describes the interface implementations for the CLM land surface model. CLM (Community Land Model) is a 1-D land surface model, written in Fortran 90, developed by a grass-roots collaboration of scientists who have an interest in making a general land model available for public use. LIS currently uses CLM version 2.0. CLM version 2.0 was released in May 2002. The source code for CLM 2.0 is freely available from the National Center for Atmospheric Research (NCAR) [1]. The CLM is used as the land model for the Community Climate System Model (CCSM) (<http://www.cesm.ucar.edu/>), which includes the Community Atmosphere Model (CAM) (<http://www.cgd.ucar.edu/cms/>). CLM is executed with all forcing, parameters, dimensioning, output routines, and coupling performed by an external driver of the user's design (in this case done by LIS). CLM requires pre-processed data such as the land surface type, soil and vegetation parameters, model initialization, and atmospheric boundary conditions as input. The model applies finite-difference spatial discretization methods and a fully implicit time-integration scheme to numerically integrate the governing equations. The model subroutines apply the governing equations of the physical processes of the soil-vegetation-snowpack medium, including the surface energy balance equation, Richards' [15] equation for soil hydraulics, the diffusion equation for soil heat transfer, the energy-mass balance equation for the snowpack, and the Collatz et al. [4] formulation for the conductance of canopy transpiration.

42.1 Fortran: Module Interface clmtype (Source File: clmtype.F90)

The code in this file provides a description of the data structure containing the CLM 1-d variables. The variables specified in the data structure include:

nstep time step number

kpatch tile index

itypveg vegetation type
itypwat water type
itypprc precipitation type
isoicol color class for soil albedos
snl number of snow layers
frac_veg_nosno fraction of vegetation not covered by snow
frac_veg_nosno_alb fraction of vegetation not covered by snow
imelt flag for melting (=1), freezing (=2), not=0(new)
lakpoi flag for lakpoint (true=lake point)
do_capsnow flag to indicate snow capping (true=do sno caping)
present whether PFT is present in the current patch
lat latitude of the patch
lon longitude of the patch
dtime model timestep
zi interface level below a "z" level (m)
dz layer depth (m)
z layer thickness (m)
bsw Clapp and Hornberger "b"
watsat volumetric soil water at saturation (porosity)
hksat hydraulic conductivity at saturation (mm H₂O /s)
sucsat minimum soil suction (mm)
csol heat capacity, soil solids (J/m^{**3}/Kelvin)
tkmg thermal conductivity, soil minerals [W/m-K] (new)
tkdry thermal conductivity, dry soil (W/m/Kelvin)
tksatu thermal conductivity, saturated soil [W/m-K] (new)
rootfr fraction of roots in each soil layer
rootr effective fraction of roots in each layer
begwb water mass at the beginning of the time step
endwb water mass at the end of the time step
forc_t atmospheric temperature (K)
forc_u atmospheric wind speed in east direction (m/s)
forc_v atmospheric wind speed in north direction (m/s)
forc_q atmospheric specific humidity (kg/kg)

forc_hgt atmospheric reference height (m)
forc_hgt_u observational height of wind [m]
forc_hgt_v observational height of temperature [m]
forc_hgt_q observational height of humidity [m] (new)
forc_pbot atmospheric pressure (Pa)
forc_th atmospheric potential temperature (Kelvin)
forc_vp atmospheric vapor pressure (Pa)
forc_rho density (kg/m**3)
forc_lwrad downward infrared (longwave) radiation (W/m**2)
forc_solad direct beam radiation (vis=forc_sols , nir=forc_soll)
forc_solai diffuse radiation (vis=forc_solsd, nir=forc_solld)
forc_ch heat/moisture exchange coefficient
forc_rain rain rate [mm/s]
forc_snow snow rate [mm/s]
rssun sunlit stomatal resistance (s/m)
rsshha shaded stomatal resistance (s/m)
psnsun sunlit leaf photosynthesis (umol CO₂ /m**2/ s)
psnsha shaded leaf photosynthesis (umol CO₂ /m**2/ s)
laisun sunlit leaf area
laisha shaded leaf area
sabg solar radiation absorbed by ground (W/m**2)
sabv solar radiation absorbed by vegetation (W/m**2)
fsa solar radiation absorbed (total) (W/m**2)
taux wind stress: e-w (kg/m/s**2)
tauy wind stress: n-s (kg/m/s**2)
eflx_lwrad_out emitted infrared (longwave) radiation (W/m**2)
eflx_lwrad_net net infrared (longwave) rad (W/m**2) [+ = to atm]
eflx_sh_tot total sensible heat flux (W/m**2) [+ to atm]
eflx_sh_veg sensible heat flux from leaves (W/m**2) [+ to atm]
eflx_sh_grnd sensible heat flux from ground (W/m**2) [+ to atm]
eflx_lh_tot total latent heat flux (W/m8*2) [+ to atm]
eflx_soil_grnd soil heat flux (W/m**2) [+ = into soil]
t_veg vegetation temperature (Kelvin)

t_grnd ground temperature (Kelvin)
t_rad radiative temperature (Kelvin)
t_ref2m 2 m height surface air temperature (Kelvin)
t_soisno soil temperature (Kelvin)
qflx_infl infiltration (mm H₂O /s)
qflx_surf surface runoff (mm H₂O /s)
qflx_drain sub-surface runoff (mm H₂O /s)
qflx_top_soil net water input into soil from top (mm/s)
qflx_evap_soi soil evaporation (mm H₂O/s) (+ = to atm)
qflx_evap_veg vegetation evaporation (mm H₂O/s) (+ = to atm)
qflx_tran_veg vegetation transpiration (mm H₂O/s) (+ = to atm)
qflx_snomelt snow melt (mm H₂O /s)
qflx_evap_tot qflx_evap_soi + qflx_evap_veg + qflx_tran_veg
qflx_rain_grnd rain on ground after interception (mm H₂O/s) [+]
qflx_evap_grnd ground surface evaporation rate (mm H₂O/s) [+]
qflx_dew_grnd ground surface dew formation (mm H₂O /s) [+]
qflx_sub_snow sublimation rate from snow pack (mm H₂O /s) [+]
qflx_dew_snow surface dew added to snow pack (mm H₂O /s) [+]
qflx_snowcap excess precipitation due to snow capping (mm H₂O /s) [+]
qflx_qrgwl qflx_surf at glaciers, wetlands, lakes
h2osno snow water (mm H₂O)
h2ocan canopy water (mm H₂O)
h2osoi_liq liquid water (kg/m²)
h2osoi_ice ice lens (kg/m²)
h2osoi_vol volumetric soil water (0_j=h2osoi_vol_j=watsat) [m³/m³]
snowdp snow height (m)
snowage non dimensional snow age [-]
h2osno_old snow mass for previous time step (kg/m²) (new)
frac_sno fraction of ground covered by snow (0 to 1)
frac_iceold fraction of ice relative to the total water (new)
eff_porosity effective porosity = porosity - vol_ice
parsun average absorbed PAR for sunlit leaves (W/m**2)
albgrd ground albedo (direct)

albgri ground albedo (diffuse)
fabd flux absorbed by veg per unit direct flux
fabi flux absorbed by veg per unit diffuse flux
ftdd down direct flux below veg per unit dir flx
ftid down diffuse flux below veg per unit dir flx
ftii down diffuse flux below veg per unit dif flx
fsun sunlit fraction of canopy
surfalb instantaneous all-wave surface albedo
snoalb instantaneous all-wave snow albedo
hbot canopy bottom (m)
htop canopy top (m)
tlai one-sided leaf area index, no burying by snow
tsai one-sided stem area index, no burying by snow
elai one-sided leaf area index with burying by snow
esai one-sided stem area index with burying by snow
fwet fraction of canopy that is wet (0 to 1)
fdry fraction of foliage that is green and dry [-] (new)
annpsn annual photosynthesis (umol CO₂ /m^{**2})
annpsnspot annual potential photosynthesis (same units)
wf soil water as frac. of whc for top 0.5 m
z0mr ratio of momentum roughness length to canopy top height [-]
z0m momentum roughness length [m]
displar ratio of displacement height to canopy top height [-]
displa displacement height [m]
dleaf leaf dimension [m]
xl pft_varcon leaf/stem orientation index
rhol pft_varcon leaf reflectance : 1=vis, 2=nir
rhos pft_varcon stem reflectance : 1=vis, 2=nir
taul pft_varcon leaf transmittance: 1=vis, 2=nir
taus pft_varcon stem transmittance: 1=vis, 2=nir
qe25 quantum efficiency at 25c (umol co₂ / umol photon)
vcmx25 maximum rate of carboxylation at 25c (umol co₂/m^{**2}/s)
mp slope for conductance-to-photosynthesis relationship

c3psn photosynthetic pathway: 0. = c4, 1. = c3
totfsa solar absorbed solar radiation [W/m²]
toteflx_lwrad_net net longwave radiation [W/m²]
toteflx_lh_tot total latent heat flux [W/m²]
toteflx_sh_tot total sensible heat flux [W/m²]
toteflx_soil_grnd ground heat flux [W/m²]
toqflux_snomelt snowmelt heat flux [W/m²]
totrain accumulation of rain [mm]
totsnow accumulation of snow [mm]
totqflux_evap total evaporation [mm]
totqflux_surf surface runoff [mm]
totqflux_drain subsurface runoff [mm]
totqflux_ecanop interception evaporation [W/m²]
totqflux_tran_veg Total vegetation transpiration
totqflux_evap_grnd Total ground surface evaporation
totqflux_sub_snow Total sublimation rate from snow pack
acond aerodynamic conductance
soilmtc_prev Total column soil moisture for the prev.timestep
h2osno_prev Total column snow water equivalent for the prev.timestep

USES:

```
use precision, only : r8
use clm_varpar, only : nlevsoi, nlevsno, nlevlak, numrad, nvoc, ndst
```

42.2 Fortran: Module Interface `clm_varder` (Source File: `clm_varder.F90`)

This module provides the definition of derived data type used to control the operation of CLM. It also provides the entry method for the initialization of CLM-specific variables. The derived data type `clm_struct` includes the variables that specify the runtime options and other control variables as described below:

clm2_rfile name of the CLM restart file
clm2_vfile name of the CLM vegetation parameter lookup table
clm2_chtfile name of the CLM canopy heights lookup table
count variable to keep track of the number of timesteps before an output
clm2open variable to keep track of opened files
numout number of output times

clm2_ism initial soil moisture for a cold start run
clm2_it initial soil temperature for a cold start run
clm2_iscv initial snow mass for a cold start run
outInterval output writing interval
rstInterval restart writing interval
clm CLM specific variables

USES:

```
use clmtype
implicit none
```

PUBLIC MEMBER FUNCTIONS:

```
public :: clm_varder_ini
```

PUBLIC TYPES:

```
public :: clm_struct
```

42.2.1 clm_varder_ini (Source File: **clm_varder.F90**)

INTERFACE:

```
subroutine clm_varder_ini()
```

USES:

```
use spmdMod, only : iam, masterproc
use infnan
use clm_varcon
use pft_varcon
use shr_orb_mod
use lis_logmod, only : logunit
use lisdrv_module, only : lis
```

DESCRIPTION:

This routine creates the datatypes and allocates memory for CLM-specific variables. It also invokes the routine to read the runtime specific options for CLM from the configuration file.

The routines invoked are:

readclm2crd (42.3.1)
reads the runtime options for CLM

clm_varder_init (42.2.2)
set initial values to CLM variables

iniTimeConst (42.3.3)
initialize time invariant parameters

42.2.2 clm_varder_init (Source File: **clm_varder.F90**)

INTERFACE:

```
subroutine clm_varder_init(n)
```

USES:

```
use lisdrv_module, only : lis
use infnan
```

DESCRIPTION:

Initializes clm variables

The arguments are:

n index of the nest

42.3 Fortran: Module Interface atmdrvMod (Source File: **atmdrvMod.F90**)

This module defines the variables and routines to enable the interaction with an atmospheric component

USES:

```
use precision
use shr_const_mod, only : SHR_CONST_TKFRZ,SHR_CONST_PSTD
use spmdMod      , only : masterproc
```

42.3.1 atmdrv (Source File: **atmdrvMod.F90**)

INTERFACE:

```
subroutine atmdrv(n, k, forcing)
```

USES:

```
use infnan
use clm_varder
use clm_varcon , only : rair, cpair, po2, pco2, tcrit, tfrz
use lisdrv_module, only : lis

implicit none
```

ARGUMENTS:

```

integer, intent(in) :: n
integer              :: k
real                 :: forcing(lis%nf)

```

DESCRIPTION:

read atmospheric data

Method: This code reads in atmospheric fields from an input file and generates the required atmospheric forcing.

Possible atmospheric fields:

Name	Description	Required/Optional
TBOT	temperature (K)	Required
WIND	wind:sqrt(u**2+v**2) (m/s)	Required
QBOT	specific humidity (kg/kg)	Required
Tdew	dewpoint temperature (K)	Alternative to Q
RH	relative humidity (percent)	Alternative to Q
ZBOT	reference height (m)	optional
PSRF	surface pressure (Pa)	optional
FSDS	total incident solar radiation (W/m**2)	Required
FSDSdir	direct incident solar radiation (W/m**2)	optional (replaces FSDS)
FSDSdif	diffuse incident solar rad (W/m**2)	optional (replaces FSDS)
FLDS	incident longwave radiation (W/m**2)	optional
PRECTmms	total precipitation (mm H2O / sec)	Required
PRECCmms	convective precipitation (mm H2O / sec)	optional (replaces PRECT)
PRECLmms	large-scale precipitation (mm H2O / sec)	optional (replaces PRECT)

The arguments are:

n index of the nest

k index of the tile

forcing input forcing array from LIS

!ROUTINE : readclm2crd

REVISION HISTORY:

14 Oct 2003; Sujay Kumar, Initial Code

INTERFACE:

```
subroutine readclm2crd()
```

USES:

```

use lis_logmod, only : logunit
use lisdrv_module, only : lis, config_lis
use LIS_ConfigMod
use clm_yarder, only : clm_struc

```

DESCRIPTION:

This routine reads the options specific to CLM from the LIS configuration file.

The routines invoked are:

LIS_ConfigGetAttribute (5.7.5)

read the specified attribute

LIS_ConfigFindLabel (5.7.8)

find the specified label to read the attribute

42.3.2 clm2_setup (Source File: clm2_setup.F90)

REVISION HISTORY:

20 Jan 2003 Sujay Kumar Initial Specification

INTERFACE:

```
subroutine clm2_setup()
```

USES:

```
use lisdrv_module, only: lis
use spmdMod
use listime_mgr
use clm_varder
use clm_varcon, only: eccen, obliqr, lambm0 , mvelpp
use lis_logmod, only :logunit
```

DESCRIPTION:

This routine is the entry point to set up the parameters required for CLM. The method invokes the routine to call time variant parameters.

The routines invoked are:

canhtset (42.3.4)

reads the canopy height information

iniTimeVar (42.3.4)

initialize time variant parameters

42.3.3 iniTimeConst (Source File: iniTimeConst.F90)

INTERFACE:

```
subroutine iniTimeConst (n)
```

USES:

```
use precision
use infnan
use clm_varder
use clm_varpar , only : nlevsoi, nlevlak
use clm_varcon , only : istsoil, istice, istdlak, istslak, istwet, spval
use pft_varcon , only : ncorn, nwheat, roota_par, rootb_par, &
                      z0mr, displar, dleaf, rhol, rhos, taul, taus, xl, &
```

```

        qe25, vcmx25, mp, c3psn
use clm_varsur , only : zlak, dzlak, zsoi, dzsoi, zisoi
use shr_const_mod, only : SHR_CONST_PI
use spmdMod      , only : masterproc
use lisdrv_module, only : lis, lisdom
use soils_module, only  : lis_soils
use lis_logmod, only   : logunit

implicit none

```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine initializes the time invariant CLM variables

The arguments are:

n index of the nest

42.3.4 iniTimeVar (Source File: iniTimeVar.F90)

INTERFACE:

```
subroutine iniTimeVar (n, eccen, obliqr, lambm0 , mvelpp)
```

USES:

```

use precision
use lisdrv_module, only : lis
use clm_varder
use clm_varcon , only : bdsno, istice, istwet, &
    istsoil, denice, denh2o, tfrz, spval, doalb
use shr_sys_mod , only : shr_sys_abort
use spmdMod      , only : masterproc
use listime_mgr, only : get_nstep, get_curr_calday
use lis_logmod, only : logunit

implicit none

```

ARGUMENTS:

```

integer , intent(in) :: n
real(r8), intent(in) :: eccen
real(r8), intent(in) :: obliqr
real(r8), intent(in) :: lambm0
real(r8), intent(in) :: mvelpp

```

DESCRIPTION:

Initialize the following time varying variables:

water : h2osno, h2ocan, h2osoi_liq, h2osoi_ice, h2osoi_vol
snow : snowdp, snowage, snl, dz, z, zi

temperature: t_soisno, t_veg, t_grnd

Note - h2osoi_vol is needed by clm_soilalb -this is not needed on restart since it is computed before the soil albedo computation is called

Note - remaining variables are initialized by calls to ecosystem dynamics and albedo subroutines.
The arguments are:

n index of the nest

eccen Earth's orbital eccentricity

obliqr Earth's obliquity in radians

lambm0 Mean longitude of perihelion at the vernal equinox (radians)

mvelpp Earth's moving vernal equinox long. of perihelion + pi (radians)

!ROUTINE:canhtset

REVISION HISTORY:

15 Nov 2002: Jon Gottschalck; Initial code

INTERFACE:

subroutine canhtset (n)

USES:

```
use clm_varder          ! CLM2 tile variables
use clm_varpar , only : maxpatch
use lisdrv_module, only : lis
```

implicit none

ARGUMENTS:

integer, intent(in) :: n

DESCRIPTION:

This subroutine reads in canopy height information into CLM2. Expects a file with the top and bottom canopy heights for each veg type.

42.3.5 clm2_dynsetup (Source File: clm2_dynsetup.F90)

REVISION HISTORY:

20 Jan 2003 Sujay Kumar Initial Specification

INTERFACE:

subroutine clm2_dynsetup(n)

implicit none

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine sets up the time-dependent variables in CLM. The routine invokes methods to read the LAI data

42.3.6 clm2_main (Source File: clm2_main.F90)

INTERFACE:

```
subroutine clm2_main (n)
```

USES:

```
use precision
use clm_varder
use clm_varcon      , only : doalb, eccen, obliqr, lambm0, mvelpp, &
    denh2o, denice, hvap, hsub, hfus, istwet
use lisdrv_module, only : lis
use listime_mgr   , only : get_step_size, get_curr_calday, get_curr_date, get_nstep
#ifndef (defined RTM)
    use RtmMod      , only : Rtmriverflux
#endif
#ifndef (defined SPMD)
    use spmdMod     , only : masterproc, npes
    use mpishorthand , only : mpir8, mpilog, mpicom
#else
    use spmdMod     , only : masterproc
#endif
#ifndef (defined COUP_CSM)
    use clm_csmMod   , only : csm_dosndrcv, csm_recv, csm_send, csm_flxave, &
        dorecv, dosend, csmstop_now
#endif
    use shr_sys_mod   , only : shr_sys_flush

    implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This is the entry point for calling the CLM physics. This routine calls various physics routines for CLM that performs the land surface computations, to solve for water and energy equations. For a more detailed documentation of CLM, please contact the NCAR repository

Calling sequence:

```
-> loop over patch points calling for each patch point:
-> Hydrology1      canopy interception and precip on ground
-> Biogeophysics1  leaf temperature and surface fluxes
-> Biogeophysics_Lake lake temperature and surface fluxes
-> Biogeophysics2  soil/snow and ground temp and update surface fluxes
-> Hydrology2      surface and soil hydrology
```

```

-> Hydrology_Lake      lake hydrology
-> Biogeochemistry    surface biogeochemical fluxes (LSM)
-> EcosystemDyn:      ecosystem dynamics: phenology, vegetation,
                      soil carbon
-> SurfaceAlbedo:    albedos for next time step
  -> SnowAlbedo:      snow albedos: direct beam
  -> SnowAlbedo:      snow albedos: diffuse
  -> SoilAlbedo:      soil/lake albedos
  -> TwoStream:       absorbed, reflected, transmitted
                      solar fluxes (vis dir)
  -> TwoStream:       absorbed, reflected, transmitted
                      solar fluxes (vis dif)
  -> TwoStream:       absorbed, reflected, transmitted
                      solar fluxes (nir dir)
  -> TwoStream:       absorbed, reflected, transmitted
                      solar fluxes (nir dif)
-> BalanceCheck        check for errors in energy and water balances

```

The arguments are:

n index of the nest

42.3.7 clm2_output (Source File: clm2_output.F90)

INTERFACE:

```
subroutine clm2_output(n)
```

USES:

```

use lisdrv_module, only : lis
use clm_varder, only : clm_struc
use spmdMod, only : masterproc, npes
use lis_fileIOMod, only : create_output_directory, &
                         create_output_filename,  &
                         create_stats_filename
implicit none

```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This subroutine sets up methods to write model output from CLM A timestamped filename and directory are created first. The routine supports model output in three data formats: Binary, Grib and NETCDF.
The arguments are:

n index of the nest

The routines invoked are:

create_output_directory (5.23.7)
creates a time stamped output directory

create_output_filename (5.23.8)
creates a time stamped output filename

create_stats_filename (5.23.10)
creates a stats filename

clm2_binout (42.3.8)
routine to write CLM model output in binary format

clm2_gribout (42.3.9)
routine to write CLM model output in grib format

clm2_totint (42.3.10)
resets the time averaged variables

42.3.8 clm2_binout (Source File: **clm2_binout.F90**)

REVISION HISTORY:

02 Dec 2003: Sujay Kumar; Initial Version

INTERFACE:

```
subroutine clm2_binout(ftn,ftn_stats,n)
```

USES:

```
use lisdrv_module, only : lis,metadata_output
use drv_output_mod, only : drv_writevar_bin
use clm_varcon, only : denh2o, denice, hvap, hsub, hfus, istwet
use clm_varpar, only : nlevsoi
use clm_varder, only : clm_struc

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine writes the specified CLM variables into a binary output file.
The arguments are:

n index of the nest

ftn unit number for the model output file

ftn_stats unit number for the stats file

The routines invoked are:

drv_writevar_bin (5.21.2)
routine to write a variable to a binary output file

42.3.9 clm2_gribout (Source File: clm2_gribout.F90)

REVISION HISTORY:

02 Dec 2003: Sujay Kumar; Initial Version

INTERFACE:

```
subroutine clm2_gribout(ftn,n)
```

USES:

```
use lis_module
use lisdrv_module, only : lis,lisdom
use drv_output_mod, only : drv_writevar_grib
use clm_varcon, only : denh2o, denice, hvap, hsub, hfus, istwet
use clm_varpar, only : nlevsoi
use clm_varder, only : clm_struc
use listime_mgr, only : tick

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine writes the specified CLM variables into a grib output file.
The arguments are:

n index of the nest

ftn unit number for the model output file

The routines invoked are:

drv_writevar_grib (5.21.5)
routine to write a variable to a grib output file

42.3.10 clm2_totinit (Source File: clm2_totinit.F90)

REVISION HISTORY:

14 Jun 2002 Sujay Kumar Initial Specification

INTERFACE:

```
subroutine clm2_totinit(n)
```

USES:

```
use lisdrv_module, only : lis
use clm_varder

implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This routine resets the time-averaged CLM variables after a model output. The routine is called after a model output call.

42.3.11 clm2wrst (Source File: clm2wrst.F90)

REVISION HISTORY:

```
20 Jan 2003; Sujay Kumar Initial Specification  
26 Aug 2004; James Geiger, Added support for GrADS-DODS based and MPI based  
parallel simulations.
```

INTERFACE:

```
subroutine clm2wrst(n)
```

USES:

```
use spmdMod, only : masterproc, npes  
use lisdrv_module, only : lis  
use lis_logmod, only : logunit  
use lis_fileIOMod, only : create_output_directory, &  
                         create_restart_filename  
use clm_varder, only : clm_struc  
  
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
```

DESCRIPTION:

This program writes restart files for CLM. This includes all relevant water/energy storage and tile information
The routines invoked are:

create_output_directory (5.23.7)

creates a timestamped directory for the restart files

create_restart_filename (5.23.9)

generates a timestamped restart filename

clm_dump_restart (42.3.12)

writes the CLM variables into the restart file reads a variable from the restart file

42.3.12 clm_dump_restart (Source File: clm2wrst.F90)

INTERFACE:

```
subroutine clm_dump_restart(n, ftn)
USES:
use clm_varder
use listime_mgr
use drv_output_mod, only : drv_writevar_restart
implicit none
```

ARGUMENTS:

```
integer, intent(in) :: n
integer, intent(in) :: ftn
```

DESCRIPTION:

This routine gathers the necessary restart variables and performs the actual write statements to create the restart files.

The arguments are:

n index of the nest

ftn unit number for the restart file

The routines invoked are:

drv_writevar_restart (5.21.3)

 writes a variable to the restart file

42.3.13 clm2_restart (Source File: **clm2_restart.F90**)

REVISION HISTORY:

20 Jan 2003; Sujay Kumar Initial Specification

INTERFACE:

```
subroutine clm2_restart()
```

USES:

```
use spmdMod, only : npes
use lisdrv_module, only : lis
use clm_varder
use listime_mgr
use lis_logmod, only : logunit
use drv_output_mod, only : drv_readvar_restart
```

DESCRIPTION:

This program reads restart files for CLM. This includes all relevant water/energy storages and tile information.

The routines invoked are:

drv_readvar_restart (5.21.4)

 reads a variable from the restart file

42.3.14 clm2_finalize (Source File: clm2_finalize.F90)

REVISION HISTORY:

26 Oct 2005: Sujay Kumar; Initial Specification

INTERFACE:

```
subroutine clm2_finalize()
```

USES:

```
use lisdrv_module, only : lis
use clm_varder
```

DESCRIPTION:

This routine cleans up the allocated memory structures in CLM

References

- [1] CLM. <http://www.cgd.ucar.edu/tss/clm>.
- [2] NOAH. <ftp://ftp.ncep.noaa.gov/pub/gcp/ldas/noahlsm/>.
- [3] M.-D. Chous. A solar radiation model for use in climate studies. *Journal of Atmospheric Science*, 49:762–772, 1992.
- [4] G. J. Collatz, C Grivet, J. T. Ball, and J. A. Berry. Physiological and environmental regulation of stomatal conductance: Photosynthesis and transpiration: A model that includes a laminar boundary layer. *Agric. For. Meteorol.*, 5:107–136, 1991.
- [5] B. Cosgrove, D. Lohmann, K. E. Mitchell, P. R. Houser, E. F. Wood, J. C. Schaake, A. Robock, C. Marshall, J. Sheffield, Q. Duan, L. Luo, R. W. Higgins, R. T. Pinker, J. D. Tarpley, and J. Meng. Real-time and retrospective forcing in the north american land data assimilation system (NLDAS) project. *Journal of Geophysical Research*, 108, 2003.
- [6] J. C. Derber, D.F. Parrish, and S.J. Lord. The new global operational analysis system at the national meteorological center. *Weather and Forecasting*, 6:538–547, 1991.
- [7] Chen. F., Mitchell. K., Schaake. J, Xue. J, Pan. H, Koren. V., Ek. M Duan, and A. Betts. Modeling of land-surface evaporation by four schemes and comparison with fife observations. *J. Geophys. Res.*, 101(D3):7251–7268, 1996.
- [8] T.M. Hamill, R.P.D. D'Entremont, and J.T. Bunting. A description of the air force real-time nephanalysis model. *Weather Forecasting*, 7:288–306, 1992.
- [9] P. G. Jarvis. The interpretation of leaf water potential and stomatal conductance found in canopies of the field. *Phil. Trans. R. Soc.*, 273:593–610, 1976.
- [10] S. V. Kumar, C. D. Peters-Lidard, Y. Tian, P. R. Houser, J. Geiger, S. Olden, L. Lighty, J. L. Eastman, B. Doty, P. Dirmeyer, J. Adams, K. Mitchell, E. F. Wood, , and J. Sheffield. Land information system-an interoperable framework for high resolution land surface modeling. *Environmental Modelinng and Software*, 2006.

- [11] K. E. Mitchell, D. Lohmann, P. R. Houser, J. C. Wood, E. F. Schaake, A. Robock, B. Cosgrove, J. Sheffield, Q. Duan, L. Luo, W. R. Higgins, R. T. Pinker, J. D. Tarpley, D. P. Lettenmaier, C. H. Marshall, J. K. Entin, M. Pan, W. Shi, V. Koren, J. Meng, B. H. Ramsay, and A. A. Bailey. The Multi-institution North American Land Data Assimilation system (NLDAS): Utilization of multiple GCIP products and partners in a continental distributed hydrological modeling system. *Journal of Geophysical Research*, 109:DOI:10.1029/2003JD003823, 2004.
- [12] J.-J. Morcrette, L. Smith, and Y. Fouquart. Pressure and temperature dependence of the absorption in longwave radiation parameterizations. *Beiträge zur Physik der Atmosphäre*, 59:455–469, 1986.
- [13] J.J. Morcrette. Radiation and cloud radiative properties in the ECMWF operational weather forecast model. *Journal of Geophysical Research*, 96D:9121–9132, 1991.
- [14] C. D. Peters-Lidard, S. V. Kumar, Y. Tian, J. L. Eastman, and P. R. Houser. Global urban-scale land-atmosphere modeling with the land information system. In *Symposium on Planning, Nowcasting, and Forecasting in the Urban Zone, 84th AMS Annual Meeting*, Seattle, WA., January 2004.
- [15] L. A. Richards. Capillary conduction of liquids in porous media. *Physics*, 1:318–333, 1931.
- [16] M. Rodell, P. R. Houser, U. Jambor, J. Gottschalck, K. Mitchell, C.-J. Meng., K. Arsenault, B. Cosgrove, J. Radakovich, M. Bosilovich, J. K. Entin, J. P. Walker, D. Lohmann, and D. Toll. The Global Land Data Assimilation System. *Bulletin of the American Meteorological Society*, 85(3):381–394, 2004.
- [17] E. Rogers, T. L. Black, D. G. Deaven, G. J. DiMego, Q. Zhao, M. Baldwin, N. W. Junker, and Y. Lin. Changes to the operational “early” eta analysis/forecast system at the national centers of environmental prediction. *Wea. Forecasting*, 11:391–413, 1996.
- [18] L.S. Rothman, R.R. Gamache, A. Barbe, A. Goldman, J.R. Gillis, L.R. Brown, R.A. Roth, J.-M. Flaud, and C. Camy-Peyret. Afgl atmospheric absorption line parameters compilation. *Applied Optics*, 22:2247–2256, 1982.
- [19] M. D. Schwarzkopf and S. B.Fels. The simplified exchange method revisited: An accurate rapid method for computation of infrared cooling rates and fluxes. *Journal of Geophysical Research*, 96:9075–9096, 1991.
- [20] R. Shapiro. A simple model for the calculation of the flux of direct and diffuse solar radiation through the atmosphere. Technical Report 35, AFGL Scientific Report, 1987.